*aerospace*

MDPI

*Review*

# Major Contemporary Optimal Control Methods: Current Status and Future Prospects

**Dongyang Fang [1,3], Ruizhi He [2,4], Guochuan Yu [1], Bingyi Wei [1,5] and Ying Nan [1,*]**

[1] College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China; fdybit@126.com (D.F.)
[2] State Key Laboratory of High-Efficiency Reusable Aerospace Transportation Technology, Changsha 410073, China
[3] Shanghai Electro-Mechanical Engineering Institute, Shanghai 201109, China
[4] College of Aerospace Science and Engineering, National University of Defense Technology, Changsha 410073, China
[5] Beijing Aerospace Automatic Control Research Institute, Beijing 100854, China
* Correspondence: nanying@nuaa.edu.cn

**Abstract**

Complex aerospace missions place increasing demands on guidance accuracy, resource efficiency, and autonomous decision-making. These requirements have extended the role of optimal control from offline trajectory design to onboard guidance under uncertainty and computational constraints. This paper reviews the principles, applications, and development of major optimal control methods for aerospace guidance. A common mathematical formulation is first introduced to connect vehicle dynamics, mission constraints, and performance objectives. Classical trajectory optimization methods are then reviewed, including indirect and direct methods, pseudospectral methods, and global search techniques. The discussion subsequently examines receding-horizon guidance, fast local solvers, and convexification methods for online trajectory optimization. Robust, stochastic, adaptive, and fault-aware approaches are considered for guidance under uncertain and changing conditions. Particular attention is given to learning-based control mappings, learning-enhanced numerical optimization, and reinforcement learning. Integrated guidance and control, distributed coordination, and multistage planning are also discussed in relation to system-level autonomy. Representative aerospace applications are used to compare the main advantages, limitations, and application conditions of these methods. Finally, current challenges and future research directions are outlined for coupled-dynamics planning, computation-constrained guidance, uncertainty and fault adaptation, and cooperative multistage missions.

**Keywords:** aerospace optimal guidance; optimal control; autonomous guidance and control

Check for updates

Academic Editor: Andrea Colagrossi

Received: 19 September 2026
Revised: 13 November 2026
Accepted: 15 November 2026
Published: 18 November 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

## 1. Introduction

### *1.1. Overview*

Aerospace missions include powered landing [1], multi-vehicle cooperation [2], orbital transfer [3], rendezvous and docking [4], and vehicle reentry [5]. These missions require vehicles to operate under increasingly complex dynamics, environmental uncertainties, and mission constraints [6–10]. Reaching a prescribed terminal state is therefore only part of the guidance task. Limited fuel, finite control authority, and high mission costs also require improvements in flight time, terminal accuracy, and resource utilization. These

requirements have stimulated the development of optimal control methods and trajectory optimization algorithms for aerospace vehicles [9].

Classical trajectory optimization primarily considers how to obtain an optimal state and control sequence under specified dynamics and mission conditions. Indirect methods derive the necessary conditions for optimality and provide insight into the structure of the solution. However, their application becomes more difficult as nonlinearities, boundary conditions, and path constraints increase [11]. Direct methods address these difficulties by transcribing the continuous optimal control problem into a finite-dimensional nonlinear programming (NLP) problem. Collocation and pseudospectral techniques improve numerical accuracy and constraint handling. Metaheuristic methods offer alternative search tools for multimodal problems or when derivative information is difficult to obtain [11–13]. Together, these approaches have established an important basis for trajectory design. Nevertheless, a high-quality trajectory computed before flight may become unsuitable when the actual state departs from the prediction.

This limitation has promoted a shift from offline trajectory generation to real-time optimal guidance. Model predictive control (MPC) repeatedly updates a finite-horizon optimization problem using the latest state information, allowing guidance decisions to be corrected during flight [14]. Fast local solvers and sequential convex programming (SCP) exploit problem structure to reduce the computational burden of repeated optimization [15]. The guidance system must generate a trajectory with acceptable performance and deliver an executable command within the available computation time. Numerical solution quality, recursive feasibility, and closed-loop behavior consequently become closely connected.

Online replanning, however, still depends on the accuracy of the prediction model. Atmospheric and aerodynamic uncertainties, navigation errors, external disturbances, and execution errors can affect both trajectory performance and constraint satisfaction [7,8,10]. These difficulties have encouraged the development of robust, stochastic, and adaptive guidance methods. Robust methods account for prescribed uncertainty bounds, stochastic methods incorporate probabilistic descriptions and risk requirements, and adaptive methods use new observations to update models or control decisions [9,14]. Faults introduce an additional challenge because they may change the available control capability and make the original mission infeasible. Reliable guidance must compensate for deviations and identify when control reconfiguration or a revised mission objective is needed.

As uncertainty treatment and model fidelity improve, the computational demands of online guidance also increase. This has motivated growing interest in learning-based methods. Neural networks can approximate optimal control mappings, predict initial guesses, or replace expensive model evaluations. Reinforcement learning can obtain feedback policies through interaction with a simulated environment [16]. These approaches shift part of the online computation to offline data generation and training. Their practical value is limited by training coverage and model fidelity, especially under unfamiliar conditions. Learning-based guidance can be combined with conventional feedback, runtime assurance (RTA), and verification tools to address execution and safety requirements [4,17,18]. These combinations can retain model-based checks around a learned component.

The development of individual algorithms also reveals a broader system-level issue. A feasible guidance command may exceed the tracking capability of the inner control loop. A locally effective maneuver may compromise formation safety or later-stage reachability. Integrated guidance and control (IGC) addresses relevant couplings between guidance decisions and vehicle dynamics [14]. Distributed coordination and multistage replanning extend this concern to interactions among vehicles and successive mission phases. Optimal guidance is thus developing into a decision process in which trajectory planning, uncertainty handling, learning, and execution must work together.

Existing reviews address these developments from complementary perspectives. Chai et al. classify guidance and control methods by stability theory, optimization, and artificial intelligence [6]. Their trajectory-design review examines constrained, multi-objective, and stochastic optimization [9]. Eren et al. discuss aerospace MPC formulations and implementation [14], while Malyuta et al. emphasize lossless convexification, sequential convex programming, and MPC [15]. Building on the control-capability perspective of our previous work [19], this review follows the progression from trajectory optimization to real-time, uncertainty-aware, learning-enhanced, and system-level guidance. It organizes the five groups through eight evaluation dimensions and uses representative evidence to clarify their roles and limits. Particular attention is given to the relationships between numerical solution, feedback design, uncertainty information, and control execution. This perspective supports the selection and combination of methods for different aerospace missions.

### *1.2. Review Approach and Organization*

This narrative review covers foundational studies from the 1960s and subsequent developments identified through a Web of Science search updated in September 2026. Selection focuses on aerospace trajectory optimization, feedback guidance, uncertainty treatment, learning-assisted optimization or control, and interfaces with coordination and mission planning. Studies are included when they explain one of these methodological roles and its assumptions or application conditions. Section 2 defines the formulation and evaluation indicators. Sections 3–7 examine the five method groups, and Section 8 synthesizes their roles and limitations. Figures 1–6 illustrate the guidance architecture and its principal numerical and execution structures.

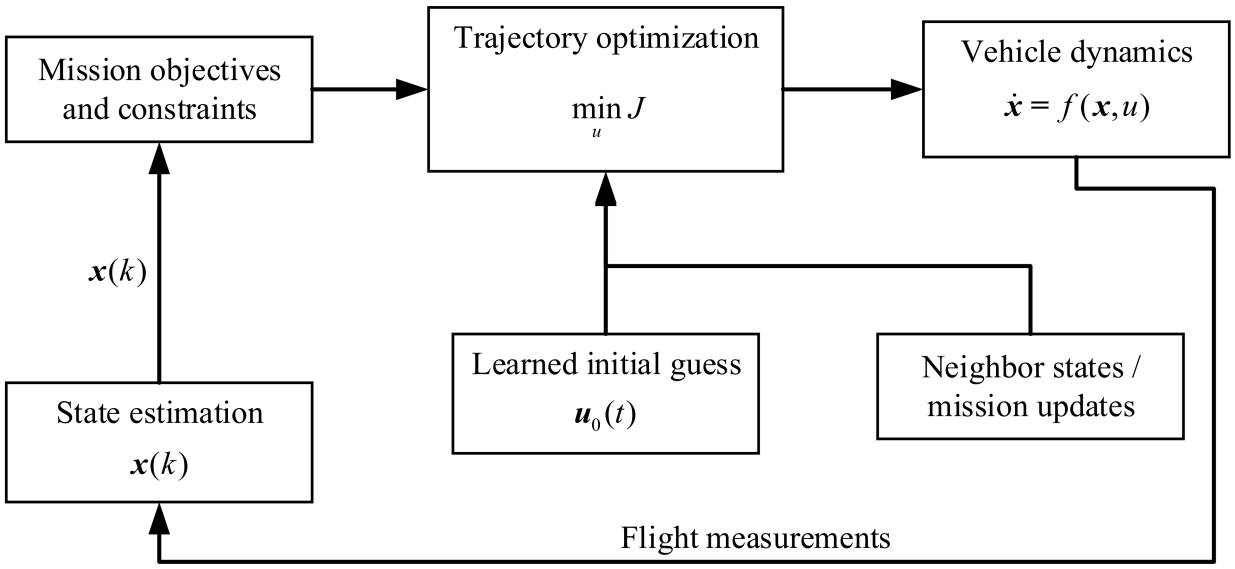



**Figure 1.** Functional architecture of optimization-based guidance. Mission objectives and constraints define the trajectory problem; state estimates close the feedback loop, while learned initial guesses and neighbor or mission updates can assist replanning. Here x is the state, u the control, f the vehicle dynamics, J a selected scalar objective or scalarization of the objectives in Section 2, t time, and k the guidance-update index. The subscript 0 labels the initial control guess. Intermediate guidance-control interfaces are omitted. Figures 2–6 detail the numerical and execution structures.

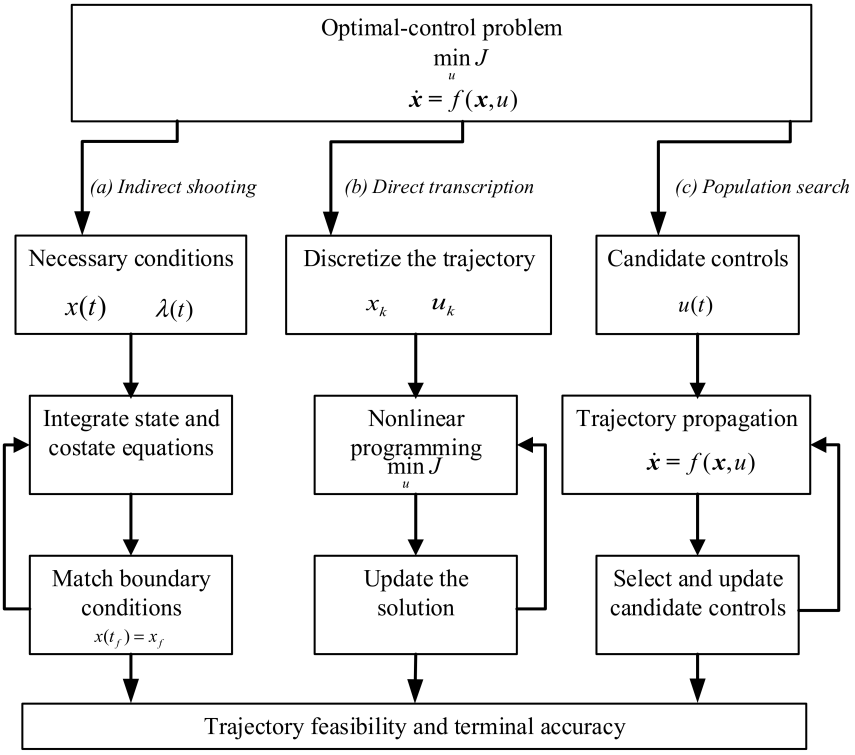



**Figure 2.** Representative routes for classical trajectory optimization. (a) Indirect methods integrate state and costate equations and adjust initial costates to meet boundary conditions. (b) Direct methods optimize discretized state and control variables. (c) Population-based search updates candidate control parameters. Each route checks the resulting trajectory against the original dynamics and constraints. Here λ denotes the costate, k indexes discretization nodes in panel (b), and the subscript f denotes the terminal time or target state. Numerical candidates have method-dependent feasibility and optimality properties.

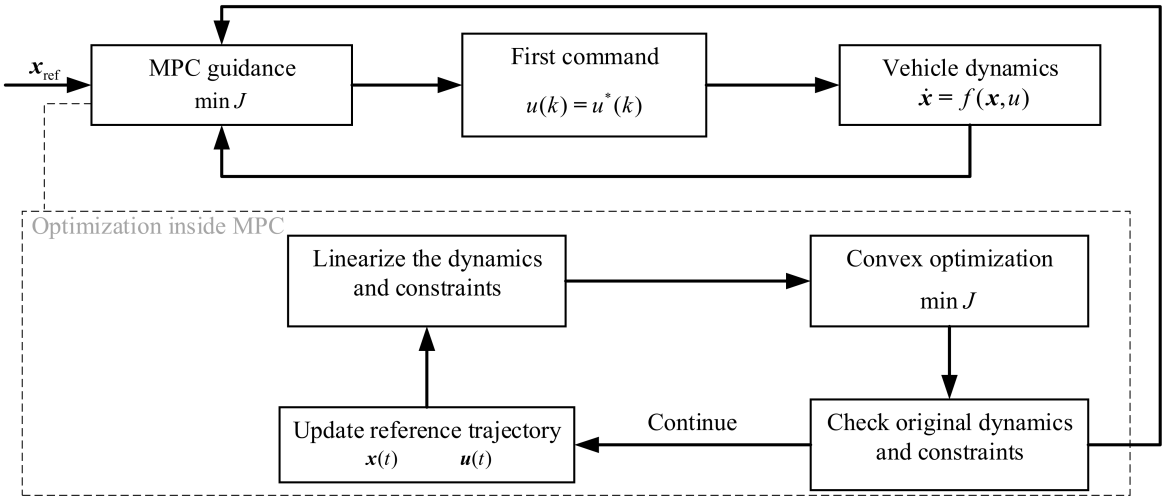



**Figure 3.** Receding-horizon guidance and a representative SCP iteration. The upper loop uses the current state to optimize a finite-horizon trajectory, applies the first optimized control, and feeds the measured state back to the next update. The dashed region shows local linearization, convex optimization, checking of the original dynamics and constraints, and reference-trajectory updates within the solver. Here k indexes guidance updates, the subscript ref denotes the reference, and the asterisk denotes an optimized control. Solver stopping rules and command acceptance depend on the implementation.

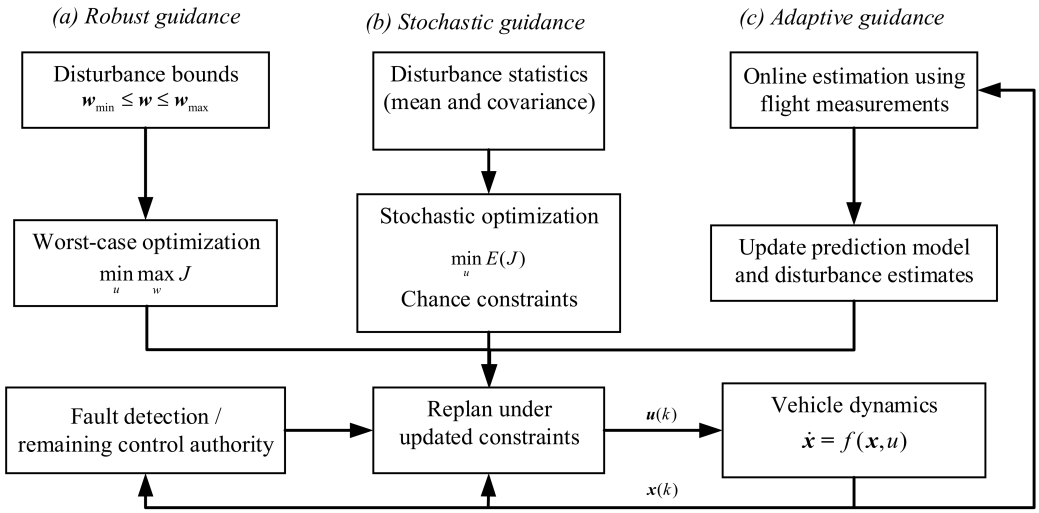



**Figure 4.** Information paths for robust, stochastic, adaptive, and fault-aware guidance. (**a**) Disturbance bounds support worst-case optimization. (**b**) Disturbance statistics support expected-cost optimization and chance constraints under specified distributional assumptions or valid moment-based bounds. (**c**) Measurements update model and disturbance estimates. Fault detection and remaining control authority inform replanning under updated constraints. The three routes are alternatives or can be combined when their assumptions are compatible. Here w denotes disturbance, the subscripts min and max denote its bounds, and E denotes expectation. The vehicle block shows the nominal dynamics.

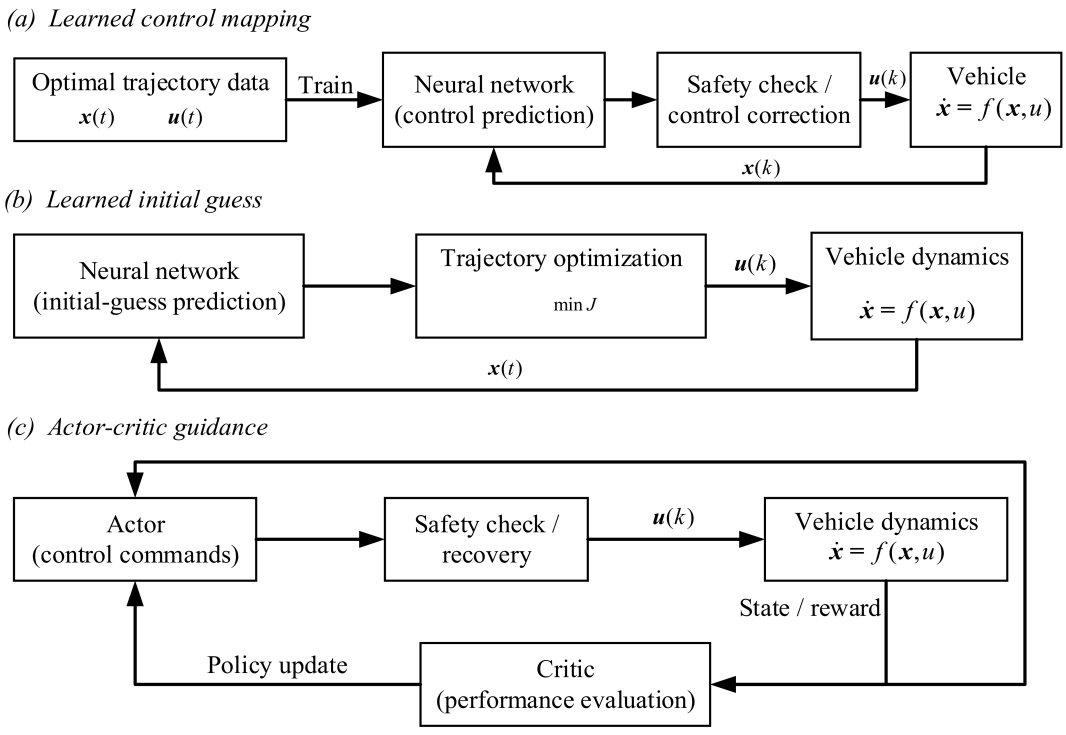



**Figure 5.** Learning-based guidance structures. (**a**) Supervised learning approximates a state-to-control mapping from optimal trajectories, with a correction or constraint-checking stage before execution. (**b**) A learned initial guess initializes numerical trajectory optimization. (**c**) Actor-critic training uses a critic to evaluate the policy and an actor to generate commands; the trained weights may remain fixed during deployment. Here x(t) and u(t) denote continuous-time state and control variables; x(k) and u(k) denote their values at guidance update k. The correction and recovery mechanisms depend on the chosen guidance architecture.

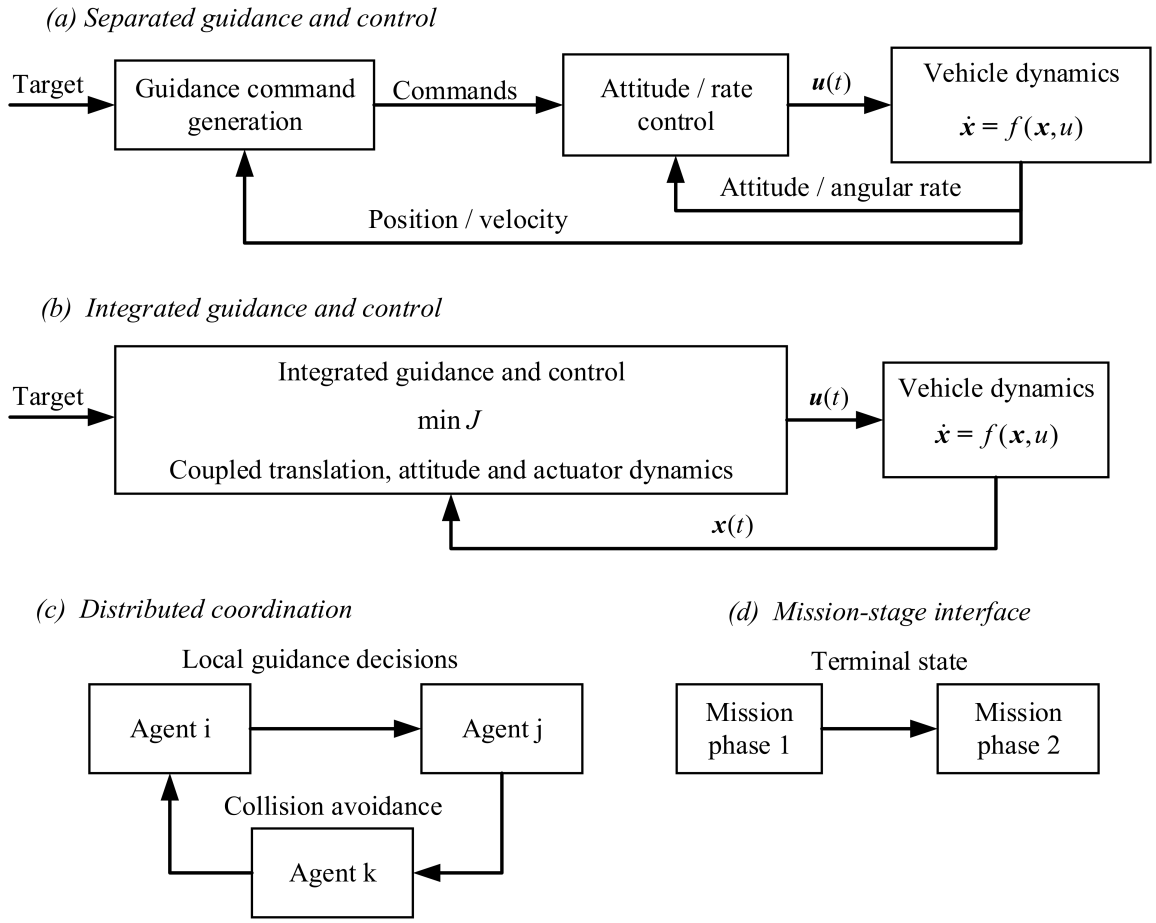



**Figure 6.** System interfaces for autonomous guidance. (**a**) Separated guidance and control use distinct motion and attitude feedback loops. (**b**) An optimization-based integrated guidance-and-control architecture accounts for coupled translation, attitude, and actuator dynamics. (**c**) A directed cycle illustrates coordination among agents i, j, and k. (**d**) A phase's terminal state initializes the next phase, whose constraints determine handoff feasibility. Interfaces and communication topology depend on the mission.

## 2. Problem Formulation and Evaluation Framework

Optimal control seeks control inputs that optimize a performance index subject to the system dynamics and constraints. Trajectory optimization computes the corresponding state and control histories, while optimal guidance uses these solutions or policies to generate commands during a mission. Online replanning extends this process by updating the solution as new state information becomes available. The formulation below describes the coupled dynamics, feasible trajectories, and performance objectives underlying these tasks [6,9,15].

Following the system representation in ref. [19], Equations (1)–(4) describe vehicle, subsystem, and environmental dynamics. Coupling is retained when it affects guidance accuracy or command execution. Examples include thermally induced solar-panel deformation [20] and the interaction between attitude and translational motion [21]. Most studies reviewed here use a single-vehicle specialization, with additional subsystem states retained only when needed. Equations (5)–(8) specify initial, terminal, and path constraints; Equations (9)–(11) define the objectives. Section 3 solves this trajectory problem, and Section 4 repeats it using updated state information. Section 5 changes the uncertainty and constraint interpretation. Section 6 learns solutions or assists their computation, while Section 7 treats inter-vehicle coupling and transitions between mission phases.

The modeled system contains $N$ top-level subsystems, denoted by $S_i$. Subsystem $i$ consists of $M_i$ second-level subsystems. Further levels are retained only when required by the mission and modeling fidelity.

Following Equation (1) in ref. [19], the $N$ top-level subsystem dynamics are written in the equivalent compact vector form:

$$\frac{d\boldsymbol{x}_i(t)}{dt} = f_i[t, \mathbf{X}(t), \boldsymbol{u}_i(t), \boldsymbol{w}_i(t), \boldsymbol{p}_i, \boldsymbol{\Xi}(t)], i = 1, \ldots, N \tag{1}$$

where the time domain of subsystem $i$ is $t \in [t_{i0}, t_{if}]$, $i = 1, 2, \ldots, N$. Here, $\boldsymbol{t}_{i0}$ and $\boldsymbol{t}_{if}$ denote the initial and final times, respectively. Let $\mathbf{X}_i(t) \in \mathbb{R}^{n_i}$ denote the state vector of subsystem $S_i$, where $n_i$ is its state dimension. $\mathbf{X}(t) = [\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_N(t)]^T$ is the global state vector formed by all subsystems. $\boldsymbol{u}_i(t) \in \mathbb{R}^{m_i}$ is the independent control variable of subsystem $i$. $\boldsymbol{w}_i(t) \in \mathbb{R}^{L_i}$ represents the motion states and disturbances of external objects or environments outside the $N$ control subsystems. $\boldsymbol{p}_i \in \mathbb{R}^{Q}$ is the vector of static parameters for subsystem $i$. $\boldsymbol{\Xi}(t)$ denotes the collection of all lower-level state variables across the subsystems and is organized schematically as the following indexed collection:

$$\boldsymbol{\Xi}(t) = \begin{bmatrix} \boldsymbol{x}_{S_1} \\ \boldsymbol{x}_{S_2} \\ \vdots \\ \boldsymbol{x}_{S_N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{S_{1,1}} & \boldsymbol{x}_{S_{1,2}} & \cdots & \boldsymbol{x}_{S_{1,M_1}} \\ \boldsymbol{x}_{S_{2,1}} & \boldsymbol{x}_{S_{2,2}} & \cdots & \boldsymbol{x}_{S_{2,M_2}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_{S_{N,1}} & \boldsymbol{x}_{S_{N,2}} & \cdots & \boldsymbol{x}_{S_{N,M_N}} \end{bmatrix} \tag{2}$$

Here, within top-level subsystem $i$, the state vector of second-level subsystem $j$ is $\boldsymbol{x}_{S_{i,j}} = \left[x_{S_{i,j,1}}, x_{S_{i,j,2}}, \ldots, x_{S_{i,j,\Phi_{i,j}}}\right]$, $j = 1, 2, \ldots, M_i$. Here, $\Phi_{i,j}$ denotes the dimension of this state vector, i.e., the number of its scalar state components: $\Phi_{i,j} = dim(\boldsymbol{x}_{S_{i,j}})$. The state definition fixes this positive integer for a given model. Each component $x_{S_{i,j,k}}$, with $k = 1, \ldots, \Phi_{i,j}$, is governed by its corresponding nonlinear differential equation. $f_i(\cdots)$ denotes the motion function of the interacting dynamic stochastic subsystems.

The array in Equation (2) indexes lower-level subsystem states; its blocks may have different dimensions. The required model resolution depends on the mission dynamics. Orbital maneuvers often permit a separation between slow orbital motion and fast attitude motion, whereas their time scales can become comparable during proximity operations [21]. Retaining the relevant coupling avoids applying a reduced model beyond the conditions supporting that reduction.

$\boldsymbol{w}_i(t)$ can represent an external state with its own dynamics. Its evolution can be described by the following differential equation:

$$\frac{\mathrm{d}\boldsymbol{w}_i(t)}{\mathrm{d}t} = h[t, \boldsymbol{w}(t), \boldsymbol{v}_i(t), \mathbf{X}(t), \boldsymbol{\Xi}(t)], i = 1, 2, \ldots, N \tag{3}$$

where the initial condition is $\boldsymbol{w}_i(t_{i0}) = \boldsymbol{w}_{i0}$. $\boldsymbol{v}_i(t)$ is the input to the external dynamics. It may represent a stochastic input or a decision input of an external object. The collection of external states and disturbances is $\boldsymbol{w}(t) = [\boldsymbol{w}_1(t), \ldots, \boldsymbol{w}_N(t)]$.

External-state dynamics describe the evolution of disturbances or interacting objects, while an uncertainty model describes imperfect knowledge of those quantities. Probabilistic propagation requires distributions and dependence assumptions for uncertain states, parameters, or inputs [22,23]. Set-based descriptions instead specify admissible bounds. For a maneuvering opponent, an interactive decision model may also be required, as in orbital pursuit-evasion [24]. The available information determines the robust or stochastic formulation used in Section 5.

For each second-level subsystem, the $\Phi_{i,j}$ scalar state variables $\boldsymbol{x}_{S_{i,j,k}}$ form its state vector. Collecting these vectors, whose dimensions may differ, gives the following block form of the internal dynamics:

$$\frac{\mathrm{d}\boldsymbol{x}_{S_i}(t)}{\mathrm{d}t} = \wp\left[t, \boldsymbol{w}(t), \boldsymbol{v}_i(t), \boldsymbol{p}_i, \mathbf{X}(t), \boldsymbol{\Xi}(t)\right], i = 1, 2, \ldots, N \tag{4}$$

Subsystem $i$ starts at time $t_{i0}$ from the initial point $O$ and evolves to the multidimensional terminal point $F$ at time $t_{if}$. The following constraints must be satisfied:

$$\boldsymbol{x}_i(t_{i0}) = \boldsymbol{x}_{i0}, \quad \boldsymbol{u}_i(t_{i0}) = \boldsymbol{u}_{i0}, \quad \boldsymbol{x}_{S_i}(t_{i0}) = \boldsymbol{x}_{S_{i0}} \tag{5}$$

The initial conditions describe the system state available when the optimization begins. If the actual mission does not prescribe the initial control input, that input need not be fixed. Online guidance should distinguish the true initial state from the navigation estimate. Estimation error can affect the predicted trajectory and terminal performance [25].

At the terminal time, the following conditions must be satisfied:

$$\begin{cases} C_1\left[t_{if}, \boldsymbol{x}_i(t_{if}), \boldsymbol{u}_i(t_{if}), \boldsymbol{x}_{S_{if}}\right] = 0 \\ C_2\left[t_{if}, \boldsymbol{x}_i(t_{if}), \boldsymbol{u}_i(t_{if}), \boldsymbol{x}_{S_{if}}\right] \leqslant 0 \end{cases} \tag{6}$$

where $\boldsymbol{x}_{S_{i,f}}$ is the lower-level system state at the terminal time.

Throughout the motion, the following bounds on the states and variables must also be satisfied:

$$g_{i,\min} \leq g_i(t, \mathbf{X}(t), \boldsymbol{u}_i(t), \boldsymbol{w}(t), \boldsymbol{\Xi}(t)) \leq g_{i,\max} \tag{7}$$

where $g_i$ is an $n_{gi}$-dimensional vector function. The control variable $\boldsymbol{u}_i(t)$ and the static parameter $\boldsymbol{p}_i$ are constrained by the sets $\Omega_U$ and $\Omega_P$, respectively:

$$\begin{aligned} \boldsymbol{u}_{i,\min} &\leqslant \boldsymbol{u}_i(t) \leqslant \boldsymbol{u}_{i,\max} \\ \boldsymbol{p}_{i,\min} &\leqslant \boldsymbol{p}_i \leqslant \boldsymbol{p}_{i,\max} \end{aligned} \tag{8}$$

The range of a control variable reflects the input that the system can physically apply. Let $\boldsymbol{p}^d$ collect the selectable components of the parameter vectors $\boldsymbol{p}_i$ and let $\Omega_{\boldsymbol{p}^d}$ denote their admissible set inherited from $\Omega_p$. For the fixed-terminal-time formulation in Equation (9), the decision variables are the control inputs and these selectable components. A free terminal time is included as an additional decision variable when the mission requires it. Nonselectable parameters, such as mass deviations or aerodynamic-coefficient errors, enter as known data or through a specified uncertainty model. If no design parameters are selected, $\boldsymbol{p}^d$ is omitted from the optimization variables.

The multi-objective performance indices of all subsystems in this large-scale system can be written in a unified form as:

$$\min_{\boldsymbol{u}(t), \boldsymbol{p}^d} \mathbf{J} = min \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \\ \vdots \\ \boldsymbol{J}_N \end{bmatrix} = min \begin{bmatrix} J_{1,1} & J_{1,2} & \cdots & J_{1,T+1} \\ J_{2,1} & J_{2,2} & \cdots & J_{2,T+1} \\ \vdots & \vdots & \ddots & \vdots \\ J_{N,1} & J_{N,2} & \cdots & J_{N,T+1} \end{bmatrix} \tag{9}$$

where the collective control input is $\boldsymbol{u}(t) = [\boldsymbol{u}_1(t), \boldsymbol{u}_2(t), \ldots, \boldsymbol{u}_N(t)]$. The costs in Equations (10) and (11) depend on these inputs and selectable parameters through the coupled trajectories. Robust and stochastic formulations additionally specify the uncertainty model, the worst-case or probabilistic objective, and the required constraint interpretation.

Equation (9) represents a multi-objective problem resolved through weighting, priority assignment, or a Pareto criterion. Fuel consumption and flight time, for example, generally compete. The selected aggregation defines the optimization objective, while the problem

structure and solution method determine whether local, global, or Pareto optimality can be established [9].

During the motion, each subsystem contains $T$ running performance indices and one terminal performance index. The running performance indices are:

$$J_{i,j}[\boldsymbol{u},\ \boldsymbol{p}^d]\ =\ \int_{t_{i0}}^{t_{if}} L_{i,j}[t,\ \mathbf{X}(t),\ \boldsymbol{u}_i(t),\ \boldsymbol{w}_i(t),\ \boldsymbol{p}_i,\boldsymbol{\Xi}(t)]\,dt,\quad j\ =\ 1,2,\ldots,T \tag{10}$$

The terminal performance index is:

$$J_{i,T+1}[\boldsymbol{u},\boldsymbol{p}^d]\ =\ \Phi_i[\boldsymbol{x}_i(t_{if}),t_{if}] \tag{11}$$

where $\Phi_i$ is the terminal cost function.

A terminal cost penalizes endpoint error, whereas a terminal constraint specifies the required tolerance. Feasibility and objective quality must therefore be evaluated separately [9].

The comparison uses eight dimensions, each linked to observable indicators below. These indicators describe what evidence is needed for a capability claim; they are not a single aggregate score. Table 1 distinguishes direct support, partial evidence, and dimensions not addressed by its cited examples. Constraint violations, operating conditions, and the scope of any guarantee qualify every dimension. Identification and control reconfiguration are supporting functions shared by several dimensions.

Evaluation Dimension 1 (Optimality): The decision variables are the control inputs $\boldsymbol{u}^*(t) = [\boldsymbol{u}_1^*(t),\ldots,\boldsymbol{u}_N^*(t)]$ and any selectable parameters $\boldsymbol{p}^{d,*}$. Over $[t_{i0},t_{if}]$, the $N$ subsystems follow Equation (1), from the initial condition in Equation (6) to the terminal condition in Equation, subject to Equations (7) and (8). The trajectory $\mathbf{X}^*(t)$ is evaluated using the scalarization or Pareto criterion in Equation (9). Indicators include objective value relative to a stated reference, stationarity or costate consistency, and primal constraint residuals. A reported gap must identify its reference and whether that reference has certified optimality. Collocation-based costate estimation provides one example [26].

Evaluation Dimension 2 (Adaptability): Adaptability concerns adjustment to changes in dynamics, objectives, constraints, or available information. Indicators include tracking error before and after a change, adaptation or settling time, and performance with and without model updating. The learning observer in ref. [27] illustrates model updating for helicopter tracking in Section 5.3; mean and peak errors measure different aspects of adaptation.

Evaluation Dimension 3 (Robustness): Robustness is assessed against a stated disturbance or uncertainty set. Indicators include worst-case or sampled tracking error, constraint-violation frequency and magnitude, and the extra control effort required. Any stability claim must specify the equilibrium, trajectory, or invariant set and the disturbance class covered [28]. Tube-MPC experiments illustrate constraint protection under bounded navigation error [29].

Supporting Function (Control Reconfiguration): Reconfiguration updates the control action $\boldsymbol{u}_i(t)$ when subsystem composition or available authority changes [30]. Recovery time, post-change feasibility, and retained mission performance distinguish this function from disturbance rejection alone.

Supporting Function (Identification): Identification estimates unknown models or parameters to support adaptation and learning. Relevant indicators include estimation error, convergence time, and sensitivity to noise. Fuzzy input-output modeling is one example [31].

Evaluation Dimension 4 (Intelligence): Learning uses data to approximate the control law $\boldsymbol{u}_i(t)$ or assist optimization. Indicators include task success, cost error against a specified

numerical reference, training effort, and performance outside the training range. Constraint-violation rates must be reported separately from task success. Learning a landing-control mapping from optimized trajectories is illustrated in ref. [32].

Evaluation Dimension 5 (Fault Tolerance): Fault tolerance concerns feasible operation after a specified loss of actuator or sensor capability. Passive schemes cover a predefined fault set; active schemes use diagnosis to reconfigure control [30]. Indicators include detection and recovery time, post-fault constraint violations, mission completion, and remaining control authority. Disturbance-rejection or nominal tracking results alone do not establish fault tolerance. Sections 5.4 and 7.4 discuss the resulting control and mission changes.

Evaluation Dimension 6 (Real-Time Capability): The control law $\boldsymbol{u}_i(t)$ must be computed and checked within the guidance update interval. Indicators include full-update latency, timing variability, deadline-miss rate, and feasible-command availability on a specified processor. Mean solver or network-inference time measures only part of this requirement. Constrained online guidance provides a computational example [33].

Evaluation Dimension 7 (Distributed Coordination): Distributed coordination is evaluated through formation or consensus error, minimum inter-vehicle separation, communication load, and performance under delay or link loss. Connectivity assumptions determine which information each vehicle can use [34]. Section 7.2 relates these indicators to coupled objectives and individual execution limits. Single-vehicle accuracy is insufficient evidence for this dimension.

Evaluation Dimension 8 (Functional Integration): Functional integration concerns the effect of jointly designing guidance, estimation, control, or planning. Indicators include tracking and terminal errors, constraint violations, control effort, and computation relative to an otherwise matched separated architecture. Observer-based partially integrated missile guidance and control provides an example [35]. Sections 7.1, 7.3 and 7.4 discuss dynamic coupling and mission interfaces; a coupled model alone does not quantify the benefit of integration.

## 3. Classical Trajectory Optimization Methods

Classical trajectory optimization solves the dynamics and constraints in Equations (1)–(8) for a selected objective from Equations (9)–(11). Most examples specialize this formulation to one vehicle. Analytical approaches derive control relations, whereas numerical methods form a boundary-value or mathematical programming problem [11]. Their solutions provide mission-design references and initial trajectories for online replanning. Figure 2 compares the main numerical routes and the checks required for the resulting trajectories.

### *3.1. Optimality Conditions and Indirect Numerical Methods*

Indirect optimal control methods use the calculus of variations and Pontryagin's maximum principle to derive necessary conditions. These form a two-point or multipoint boundary-value problem comprising state equations, costate equations, optimality conditions, and boundary conditions [11]. The costates describe the sensitivity of the performance index to the states, while the Hamiltonian links the dynamics and the cost. When the problem structure is sufficiently simple, these conditions can be further reduced to analytical or explicit guidance relations. Cherry's E-guidance [36], for example, maps the deviation between the current state and the target boundary conditions into thrust-allocation coefficients and produces executable commands with limited onboard computation. Its efficiency comes from exploiting the problem structure in advance, but its applicability is constrained by the dynamics and thrust model.

When complex dynamics make analytical solutions difficult to obtain, the focus shifts from deriving explicit laws to solving the necessary conditions numerically. The single-

shooting method guesses the unknown initial costates, integrates the state and costate equations, and updates the initial guess according to the terminal error. It retains the continuous optimal-control structure and uses relatively few variables. However, long horizons or unstable dynamics can amplify initial-value errors and make terminal matching highly sensitive. Multiple shooting [37] divides the trajectory into several intervals, integrates each segment separately, and matches adjacent endpoints. This shortens each unstable integration interval at the cost of additional internal variables and continuity constraints. Multiple shooting reorganizes the numerical propagation while retaining the same optimality conditions. In ref. [38], modified simple shooting updates the initial costates of a Pontryagin boundary-value problem for reusable launch-vehicle trajectory design; it is therefore an indirect shooting application.

Improved shooting structures still cannot eliminate the difficulty of obtaining suitable initial costates. Homotopy and continuation methods start from an easier problem and gradually vary the performance index, constraints, or smoothing parameters. Each solution initializes the next problem. In low-thrust trajectory studies [39], an energy-optimal solution is first obtained and then continued toward the fuel-optimal problem. Costate normalization, particle-swarm initial-guess search, and control-switch detection are used to improve convergence. In conceptual design studies [40], continuation is combined with symbolic derivation to generate mission-assessment results rapidly through a sequence of simplified problems. Bounding homotopy [41] and smoothing homotopy [42] further improve the continuation path for turning points and boundary-value sensitivity, respectively. When continuation is used, convergence also depends on constructing a suitable solution path from the easier problem to the target problem.

Path constraints, bang-bang control, and singular arcs make this bridge more difficult to construct. Constraint activation changes the arc structure and junction conditions, and an incorrect prediction can lead to an incorrectly formulated boundary-value problem. The generalized gradient method [43] selects combinations of state and control variables as independent optimization variables along different trajectory segments to handle state constraints and singular arcs. The Uniform Trigonometrization Method [44] uses trigonometric control parameterization and regularization, together with secant penalty terms for state path constraints, to reduce the need for explicit arc segmentation. These approaches extend the applicability of indirect methods, but accurate derivation of the necessary conditions and identification of the problem structure are still required. For a nonconvex problem, satisfying the necessary conditions identifies a candidate extremal; establishing local or global optimality requires additional analysis.

### *3.2. Direct Transcription Methods*

As mission constraints increase, deriving costate equations and arc-junction conditions one by one becomes a major burden. Direct methods therefore change the object being solved. They parameterize the control or discretize both states and controls. The dynamics, boundary conditions, and path constraints are then transcribed into a finite-dimensional NLP problem [11]. This transformation sacrifices part of the continuous analytical structure, but it allows different missions to be formulated within a more unified modeling framework.

Direct shooting represents the control with a small number of parameters and generates the state trajectory through numerical integration. It is simple to implement and uses relatively few variables. However, the integration mapping can still inherit sensitivity to the initial guess, and constraints between nodes must be checked along the continuous trajectory.

Direct collocation also treats the state values at the nodes as optimization variables and enforces the dynamics at the collocation points. Hargraves and Paris [45] used piecewise cubic interpolation for the states and linear interpolation for the controls, converting the

continuous problem into an NLP solved by sequential quadratic programming (SQP). Terminal conditions, control bounds, and path constraints can therefore be included in the same optimization model. No initial costate guess or complete sequence of constrained arcs is required. The trade-off is a substantial increase in the number of variables and constraints, while accuracy and convergence depend on the mesh, interpolation order, derivative calculation, and underlying solver.

Gradient and multiplier methods provide numerical tools for optimal control in function space and after transcription. Quasi-Newton updates approximate the Hessian or its inverse from changes in iterates and gradients, avoiding explicit second-order derivatives [46]. Conjugate-gradient methods accelerate function-space descent, with penalties or related techniques handling terminal and inequality constraints [47]. Multiplier and augmented-penalty methods alleviate the ill-conditioning associated with large penalty coefficients [48]. These developments improve the efficiency of trajectory optimization and link it to general nonlinear programming.

### *3.3. Pseudospectral Methods and Mesh Refinement*

Low-order piecewise collocation is general, but higher accuracy usually requires more mesh intervals and optimization variables. Pseudospectral methods approximate the states and controls using orthogonal polynomials and Gaussian-type nodes, and discretize the dynamics and performance index through differentiation matrices and quadrature formulas. A Chebyshev pseudospectral study [49] demonstrated the possibility of obtaining high-accuracy trajectories with a relatively low discretization order. When the trajectory is sufficiently smooth, high-order global approximation can be more efficient than uniformly refining a low-order mesh. The discretization structure improves the numerical representation while retaining a nonlinear programming problem.

Costate recovery links direct transcription to continuous optimality conditions. Ref. [26] maps the Gauss pseudospectral Karush-Kuhn-Tucker (KKT) multipliers to costates. For a scalar analytical example, 40 Gauss points and TOMLAB/SNOPT yielded maximum nodal absolute errors of approximately $9 \times 10^{-7}$, $1.23 \times 10^{-5}$, and $4.2 \times 10^{-6}$ in the state, costate, and control, respectively. Processor and execution time were not reported. This example illustrates discretization accuracy and costate consistency. Gauss, Radau, and Lobatto schemes differ in their endpoint treatment and multiplier-to-costate mappings [50], making both discretization choice and mesh-error assessment important for high-accuracy solutions.

High-order methods also have clear limitations. Control switching, kinks, or constraint activation can destroy global smoothness and reduce the efficiency of high-order polynomials on a fixed node set. Ref. [51] detects nonsmooth locations and estimates errors so that nodes can be concentrated in intervals requiring higher resolution. The variable-order Radau method [52] starts from a small mesh and uses warm starts to adjust the intervals and polynomial orders progressively. It has been applied to high-fidelity missions with coupled translational, rotational, and aerodynamic dynamics. Adaptive discretization allocates computational resources where the trajectory needs higher resolution. Error estimates guide refinement and allow accuracy to be checked iteratively.

Application studies further demonstrate the engineering significance of this discretization capability. Ref. [53] compared minimum-fuel and minimum-time objectives within the same Gauss pseudospectral framework, revealing the mission trade-off between fuel saving and increased climb time.

The development of pseudospectral methods connects convergence analysis, costate estimation, numerical software, and flight implementation [12]. Adaptive meshes and efficient nonlinear programming have extended their use to increasingly complex trajectories.

Further progress in convergence under weaker assumptions, hybrid-system treatment, and embedded computation remains important for high-fidelity online guidance.

### *3.4. Global Search Methods*

The indirect methods, direct methods, and nonlinear programming techniques discussed above usually rely on local iteration, so their results can vary with the initial guess and parameterization. Complexity results for general nonconvex programming [54] show that obtaining a rigorous global optimum efficiently for every problem encounters fundamental computational difficulties. Metaheuristics such as genetic algorithms, particle swarm optimization, ant colony optimization, and simulated annealing therefore provide another search mechanism. They reduce dependence on derivatives and local smoothness, but require evaluation of many candidate solutions in exchange for the ability to escape local minima and handle discrete or mixed variables [55].

In trajectory design, metaheuristics exploit problem structure to reduce the decision dimension or initialize local methods. A hypersonic reentry study [56] parameterized the bank angle as a curve of velocity and used constrained particle swarm optimization to search trajectories and estimate the landing footprint. A hybrid fractional-order particle-swarm and gravitational-search study [57] compared three algorithms over 100 independent runs. Mean objective values for the hybrid method, standard particle swarm optimization, and gravitational search were $9.37 \times 10^{-4}$, $9.45 \times 10^{-4}$, and $14.16 \times 10^{-4}$, respectively. Mean computation times were 66.64, 102.83, and 66.94 s on a 2.53 GHz Intel Core i3 using MATLAB 2016. The objective summarizes the study-specific trajectory criteria; it is not a universal measure of guidance accuracy. Three design variables, quasi-equilibrium glide conditions, and a separate load-management scheme underpin these results.

Classical methods address different numerical difficulties. Indirect methods exploit tractable optimality conditions, provided suitable costate guesses are available. Direct transcription accommodates complex path constraints, and pseudospectral or adaptive meshes improve resolution where needed. Population-based search explores multimodal initializations or discrete design choices, particularly when derivatives are unavailable. It can be combined with local refinement, but broad exploration does not certify global optimality. Propagating the computed control history checks dynamic consistency and constraints between nodes. The global-search examples reviewed here do not provide a common numerical benchmark for comparison with the other method groups.

Classical trajectory optimization connects mission requirements to computed state and control histories. It also provides reference trajectories, costates, sensitivities, and performance benchmarks. Its main limitation is that most computations assume a given model and initial condition, and the resulting solutions are usually obtained offline. During flight, deviations from the predicted state require the problem to be solved within the closed-loop update interval [12,58].

## 4. Real-Time Optimal Guidance Methods

When the flight state departs from its prediction, the offline trajectory may no longer meet the mission requirements. Real-time guidance updates the initial condition in Equation (5) and resolves a finite-horizon version of Equations (6)–(11). The resulting command must reach the execution layer within the guidance deadline. Receding-horizon feedback, numerical solvers, and onboard computing jointly determine whether this is possible [14]. Figure 3 distinguishes the feedback loop from the iterations inside an SCP solver.

### *4.1. Receding-Horizon and Model Predictive Guidance*

Receding-horizon control converts a single long-horizon optimization into a sequence of continuously updated short-horizon decisions. At each guidance cycle, future motion is predicted from the current measurement or state estimate. A control sequence is optimized, only its first segment is executed, and the problem is then solved again using the updated state. MPC integrates model prediction, performance optimization, and constraint handling within the same feedback framework [59]. It can track an offline optimal trajectory, or directly reallocate energy, time, or control margin according to the current mission state. MPC specifies how information enters the closed loop and how decisions are updated over a receding horizon. Its internal optimizer may still use gradient methods, dynamic programming, convex programming, or other numerical techniques.

The choice of prediction model determines the speed and accuracy of this feedback framework. Linear or linear time-varying models enable fast solution, but only approximate the true dynamics within a prescribed operating region. Nonlinear MPC retains more complete coupling and constraints, but requires nonlinear optimization at every cycle. Model fidelity should match the mission horizon, state deviation, and available onboard computation [14]. This also motivates hierarchical architectures: a slow loop uses a high-fidelity model to determine the mission-level trajectory, while a fast loop uses a local model for feedback correction.

Trajectory-tracking applications use offline trajectories as references for online correction. Ref. [60] used linear and nonlinear prediction models, respectively, to adjust the control over a finite horizon and explicitly handle path constraints. The results showed that receding-horizon optimization can maintain trajectory tracking after the state deviates from its prediction. As the mission objective shifts from tracking to decision-making, the prediction model also begins to support higher-level resource management. Ref. [61] used altitude at arrival in the target region as an economic objective, exploited forecast wind fields to manage limited flight energy, and used a lower-level linear time-varying MPC for rapid tracking. This separates long-horizon mission decisions from short-horizon control. Wind preview helps the upper-level planner improve the predicted arrival altitude at a specified landing region, while the faster lower-level controller tracks the resulting reference.

Terminal sets, terminal costs, and a local controller can support recursive feasibility and stability under the assumptions of the MPC design [59]. They make the predicted endpoint compatible with subsequent feasible control. Periodic terminal ingredients exploit orbital motion structure [62]. A longer prediction horizon captures later constraints and resource demands, but increases problem size and exposure to prediction error. Horizon selection must therefore match the computational budget and the guarantees supported by the model and terminal design.

### *4.2. Fast Local Trajectory Solvers*

MPC establishes the feedback architecture. The next question is how each update can be completed within the available time. General nonlinear programming must simultaneously handle many state, control, and constraint variables, while optimal trajectories at adjacent time instants usually differ only locally. Fast methods therefore exploit the temporal structure and sensitivity information of the previous trajectory and solve only for the necessary control corrections.

Differential dynamic programming (DDP) propagates a local quadratic approximation of the value function backward along a nominal trajectory to obtain control increments and local feedback. The nonlinear dynamics are then propagated forward to update the trajectory [63]. This stage-wise recursion avoids a dense second-order system for the full optimization problem. Rapid local convergence depends on the required smoothness

and second-order information. Its speed also introduces limitations. The quality of the nominal trajectory, local curvature, and positive-definiteness treatment can affect the search direction, while complex state constraints are difficult to incorporate naturally into the standard recursion. Ref. [64] approximated local gradients and curvature using quadrature points for three-dimensional terminal-angle and time guidance, reducing the burden of derivative computation while retaining local second-order structure.

Model predictive static programming (MPSP) uses terminal-state sensitivities to reduce trajectory correction to a static multiplier calculation. Generalized model predictive static programming (GMPSP) extends this approach. In the maneuvering-target example in ref. [65], generalized explicit guidance gave a terminal-range error of 0.2 m and terminal-angle errors of 0.42° and 2.5°. Eight GMPSP iterations reduced both angle errors below 0.01°, while the range error increased to 0.5 m. Thus, better angle matching was accompanied by a 0.3 m increase in range error in this case. The model included a first-order autopilot delay and a 50 ms simulation step. Neither processor nor wall-clock solve time was reported; the simulation step is not a measured computation time. Control-basis parameterization further reduces the online variables [66], provided the basis resolves the required control variation.

Local sensitivity methods can be combined with mathematical programming to handle more general costs and path constraints. Model predictive convex programming [33] places the state-increment-to-control-correction mapping in a convex subproblem, retaining MPSP-style dimension reduction. Sequential pseudospectral MPC [67] combines a local quadratic cost approximation with pseudospectral discretization and variational relations for midcourse correction. These approaches exploit local structure while increasing formulation flexibility. Warm starts, step control, and nonlinear propagation maintain the accuracy of successive approximations.

Local correction is particularly effective when successive guidance problems remain close and the previous trajectory provides a useful expansion point. Large state deviations or changes in active constraints can reduce this advantage. Combining sensitivity-based updates with reference-trajectory regeneration extends the usable range of the solver. Such a hierarchy reserves rapid correction for routine updates and invokes broader replanning when local feasibility or convergence deteriorates.

### *4.3. Convexification-Based Trajectory Optimization*

Local recursion is effective for exploiting a nearby trajectory, but general path constraints, input bounds, and nonconvex feasible regions still increase solution difficulty. Convex optimization provides another route. Mature convex solvers can provide more predictable computation when a lossless reformulation exists. Otherwise, reliable convex subproblems can be constructed around a reference trajectory [68].

Lossless convexification relaxes nonconvex constraints while preserving optimal solutions under specified conditions. For point-mass powered descent in uniform gravity, ref. [69] establishes exactness of a thrust-magnitude relaxation. A logarithmic mass transformation, conservative Taylor approximations, and discretization then produce a second-order cone program (SOCP). In its Mars landing example, omitting the no-subsurface constraint gives 387.9 kg fuel consumption and 72 s flight time, but permits underground flight. Enforcing that constraint gives 390.4 kg and 75 s; adding the glide-slope constraint gives 399.5 kg and 81 s. These SeDuMi/MATLAB results quantify the cost of the added geometric constraints. They are flight durations, not solver runtimes. Structural exactness of the relaxation remains distinct from the numerical approximations in the implemented problem.

Sequential convex programming (SCP) constructs and updates local convex approximations of nonlinear dynamics and constraints. Successive convexification (SCvx) uses a hard trust region and penalized virtual controls [70]. Guaranteed sequential trajectory opti-

mization (GuSTO) instead uses soft penalties for state-constraint and trust-region violations without these virtual controls [71]. The formulations differ in their feasibility-restoration mechanisms and convergence assumptions. Aerospace applications also exploit the choice of independent variable and discretization. Ref. [72] uses a monotonic range with virtual control and an adaptive trust region to improve high-altitude propagation and initialization robustness. Ref. [73] combines Legendre-Gauss-Lobatto (LGL) pseudospectral discretization with successive convexification for point-to-point planning; obstacle avoidance remains an extension of that formulation.

A lossless reformulation preserves optimal solutions only under the assumptions supporting its exactness result. Sequential convexification treats broader nonlinear dynamics and constraints through local approximations. Its practical performance is sensitive to initialization, step selection, and residuals of the original problem. Table 1 therefore reports the powered-descent fuel results separately from the SCP convergence and timing example below.

Convexification can also be applied to indirect formulations. Ref. [74] combines an improved Uniform Trigonometrization Method (UTM) with SCP for Mars-entry terminal-altitude maximization. The state-costate boundary-value problem is solved using control smoothing, virtual control, and buffer terms. Both unconstrained and path-constrained cases are studied; in the latter, path constraints enter the convex subproblems directly. Differences in the resulting control structures and terminal altitudes highlight the influence of regularization and constraint treatment on the computed extremals. Relaxation residuals and nonlinear trajectory propagation remain relevant to assessing their feasibility.

One minimum-downrange reentry study illustrates how trust-region choices can change convergence and reported computation time [75]. It tested 100 random reference trajectories with 100 discretization points, ECOS, and an AMD Ryzen 7 5800H at 3.20 GHz. Fourth-order soft and hard trust regions converged in 100% and 42% of the trials, respectively. Mean reported CPU times were 17.11 and 15.50 ms, with standard deviations of 2.25 and 1.67 ms. Mean normalized objectives were 0.26073 and 0.26065. The first-order soft formulation also achieved 100% convergence, with a mean objective of 0.27171. These results illustrate a case-specific trade-off. The timing scope and treatment of failed runs are not separately specified, so the values do not establish full-update latency or a general computational penalty. Other developments include a merit function separating objective and violation measures [76] and separate penalties for feasibility relaxation and the merit function [77].

### *4.4. Real-Time Computation and Closed-Loop Execution*

A guidance deadline applies to the complete update, including estimation, model revision, optimization, command transmission, and execution delay. Average solver time alone can conceal occasional overruns that leave commands based on outdated states. Warm starts reuse the previous trajectory, and control parameterization reduces the decision dimension. Specialized solvers exploit sparsity, while adaptive meshes concentrate nodes where additional resolution is needed. The combination sets the usable horizon and update rate [14]. Reported timing should identify the processor, numerical precision, solver tolerances, and whether it measures a subproblem, a full optimization, or the complete update.

Event-triggered replanning can reduce computation while the current trajectory remains feasible and prediction errors stay within prescribed bounds. Ref. [78] uses state deviation to trigger replanning, together with mesh reduction, trajectory truncation, and prediction of the state when the new command takes effect. These measures address both repeated computation and solution delay. A loose trigger may retain an obsolete trajectory; a

tight trigger approaches periodic high-frequency optimization. Its threshold must therefore be evaluated with the resulting tracking errors, constraint margins, and update delays.

Constraint representation also affects real-time performance and safety. Satisfying limits only at the discretization nodes does not exclude violations between nodes. For controllable linear time-invariant systems, ref. [79] represented the flat outputs with piecewise polynomials and converted continuous-interval constraints into a semidefinite program using polynomial positivity conditions. GPU parallelism was then used to process small matrix blocks. Constraint reformulation and hardware structure offer additional ways to improve real-time implementation. Its theoretical scope, however, remains limited by the system model, parameterization, and the positivity conditions used.

Real-time applicability requires deadline compliance on the intended processor and a feasible response when optimization is interrupted. Section 5.3 describes Raspberry Pi hardware-in-the-loop (HIL) tests without per-update latency, so deadline compliance cannot be inferred. Ground experiments, HIL tests, and flight operation provide different levels of implementation evidence. Table 1 reports simulations, HIL tests, ground experiments, or inference benchmarks; it does not document flight deployment of those particular implementations. Reusing a feasible trajectory, applying a local correction, or switching to backup guidance are possible recovery actions [14]. Their validity is limited by the current state and the duration of the interruption.

Real-time guidance combines feedback with numerical computation and command execution. MPC organizes repeated replanning, local solvers exploit trajectory structure, and convexification handles explicit constraints. Warm starts and event triggering reduce repeated work. Command checking and recovery are still required when the model or numerical approximation becomes inaccurate. The achievable update rate includes the time needed to estimate the state, solve the trajectory problem, and deliver the command. Robust and sampling-based MPC incorporate uncertainty into this feedback process [80,81].

## 5. Robust, Stochastic, and Adaptive Guidance Methods

Updated state estimates do not remove errors in the prediction model. Navigation, atmospheric, gravity-model, and execution errors must also be represented in guidance design [25,82,83]. In the Section 2 formulation, these errors affect the dynamics in Equations (1) and (3), the initial condition in Equation (5), and feasibility under Equations (6)–(8). Robust and stochastic methods specify which uncertainty realizations or probabilities the constraints must cover. Figure 4 links the available uncertainty and fault information to these choices.

### *5.1. Uncertainty Characterization and Robust Guidance*

A guidance system must first determine how errors propagate and grow along the trajectory. Linear covariance analysis propagates the error mean and covariance around a nominal closed-loop trajectory. It represents navigation, guidance, control, and maneuver-execution errors within one statistical framework. Ref. [25] used this approach to rapidly evaluate terminal dispersion and velocity increment, and to compare different correction-maneuver timings. This changes uncertainty from a post hoc Monte Carlo statistic into a predictive quantity that can support mission decisions. The core model, however, remains a linearization around the nominal trajectory with second-order statistics. Under strong nonlinearity, long propagation times, or clearly non-Gaussian dispersions, covariance alone may not be sufficient to characterize risk.

Polynomial chaos represents random states and parameters through orthogonal polynomial expansions. Ref. [22] uses Galerkin projection to derive deterministic equations for the expansion coefficients. Ref. [23] estimates them non-intrusively by regression or

pseudospectral collocation with an existing orbit propagator. Coordinate choice changes the sample count needed for a given accuracy. Stochastic dimension, polynomial order, and propagation duration determine the computational burden, while the assumed distributions bound the interpretation of the predicted statistics.

With credible disturbance bounds, robust optimization seeks performance and constraint satisfaction over a prescribed uncertainty set. Min-max optimal control treats the disturbance as an adversarial input and optimizes the worst-case outcome. Min-max differential dynamic programming [84] embeds this interaction in local value-function recursion, updating control and disturbance together for quadrotor-and-slung-load maneuvers. The disturbance channels, bounds, and local approximation define the range of uncertainty addressed by the resulting controller.

Another route is to preserve a region for error evolution around the predicted trajectory. Tube MPC decomposes the actual state into a nominal state and a disturbed error. A predesigned feedback law keeps the error within an invariant set, after which the nominal state and input constraints are tightened. Ref. [83] performs the feedback and tube design offline, so the online optimization mainly handles the nominal trajectory. Ref. [80] further incorporates navigation-estimation error into the tube and avoids treating the estimated state as the true state. The robustness guarantee depends on the verified error sets, feedback laws, and constraint-tightening relationships.

A planar air-bearing experiment illustrates the resource cost of robust constraint satisfaction [29]. Output-feedback tube MPC and conventional MPC used identical dynamics, weights, horizon, and physical constraints. In three navigation-error scenarios, their velocity increments were 0.3718/0.3321, 0.3862/0.3415, and 0.4564/0.3891 m/s, respectively. Tube MPC required 12%, 13%, and 17% more velocity increment, while maintaining the keep-out constraint that conventional MPC violated. The comparison thus links additional control effort to collision-avoidance margins under bounded estimation error. The prediction used a 10-step horizon with 4 s intervals. Larger disturbance and navigation-error sets increase constraint tightening and can substantially reduce the feasible region.

### *5.2. Stochastic Optimal Guidance*

When useful probabilistic information is available and the mission can specify an explicit risk level, stochastic methods allow a finer trade-off between performance and safety. Chance constraints require state or path limits to be satisfied with a specified probability, translating risk tolerance into a safety margin in the planning problem. Ref. [82] formulated keep-out zones and glide corridors as chance constraints and combined closed-loop covariance propagation with successive convexification to generate trajectories for soft landing and surface hopping. Compared with imposing constraints only on the nominal trajectory, the planner explicitly accounts for state dispersion. The conclusions, however, still depend on the assumed probability distribution, covariance approximation, and modeled error sources.

Stochastic formulations can exploit both receding-horizon constraints and local value-function approximations. Offline sampling-based stochastic MPC [81] constructs uncertain constraints in advance and uses a robust invariant terminal set to support recursive feasibility. Unscented-transform stochastic DDP [85] approximates state propagation and expected cost through representative points, then computes nominal control and feedback locally. Expected cost describes average performance, while chance constraints or other risk measures address low-probability failures. Their combination allows the controller to balance nominal efficiency and a specified risk tolerance.

Covariance can be actively shaped by feedback rather than treated only as a prediction output. Ref. [86] jointly optimizes a nominal trajectory and linear feedback using variable

transformations, covariance relaxation, and successive linearization for nonlinear dynamics. In its Earth-Mars low-thrust example, the reported objective increases from 8.898 to 13.640 km/s while the optimized terminal x-position variance decreases from $2.710 \times 10^{14}$ to $1.000 \times 10^{11}$ km$^2$. Monte Carlo propagation gives corresponding variances of $2.910 \times 10^{14}$ and $9.357 \times 10^{10}$ km$^2$. The sampled closed-loop y-position variance is $1.074 \times 10^{11}$ km$^2$ against a $1.000 \times 10^{11}$ km$^2$ target. The closed-loop Monte Carlo 95th-percentile control-effort cost is 11.388 km/s, distinct from the optimization estimate of 13.640 km/s. Ref. [87] combines trajectory and dispersion planning with in-atmosphere aerodynamic modulation and exo-atmospheric impulses. These examples illustrate how feedback trades control effort for reduced dispersion under a specified uncertainty model.

Credible bounds support set-based protection when constraints must hold throughout the modeled uncertainty set. Distributions or moments instead support probabilistic objectives and explicit risk allocation. Hybrid formulations can combine these descriptions. Table 1 distinguishes the tube-MPC constraint-protection experiments from the stochastic covariance-control example. Covariance reduction does not itself establish an empirical chance-constraint violation rate.

### *5.3. Online Updating and Adaptive Guidance*

Adaptive guidance updates the prediction model as new flight observations become available. Unknown parameters, lumped disturbances, and unmodeled dynamics can be estimated to improve subsequent control decisions. Ref. [27] combines a learning observer with MPC for helicopter tracking, while the extended state observer in ref. [35] estimates and compensates target maneuvers and model uncertainty. Estimation delay and residual error remain part of the closed-loop design. In the hardware-in-the-loop (HIL) tests of ref. [27], Raspberry Pi 3 and X-Plane were used with a 6.8 m/s wind disturbance. Adaptive-learning discrete-time MPC (DMPC) achieved mean position errors of 0.3628, 0.6530, and 0.0008 m, compared with 0.4454, 0.8427, and 1.9378 m for H-infinity control. The maximum x-position error increased from 4.0226 to 4.8695 m, illustrating the distinction between average tracking improvement and peak-error reduction. Per-update computation time was not reported.

Disturbance estimates allow predictable effects to be compensated, while robust margins protect against residual error. Disturbance-preview robust feedback MPC [88] predicts future effects from the estimated disturbance and its variation, combines the preview with robust constraints, and triggers optimization when needed. Preview errors must remain within the bounds used for protection. An actuator fault changes the admissible input set and requires a separate reconfiguration step.

Adaptation may also estimate the decision parameters of an interacting target. Ref. [24] uses relative-motion features and a sequence network to estimate an opponent's objective weights and switching times. Its tests report lower estimation error and computation time than a filtering baseline, and better estimation accuracy than a conventional network. These estimates can inform subsequent game-theoretic guidance within the opponent models, measurement conditions, and scenarios represented during training.

Periodic missions can exploit historical trajectory errors for model correction. Ref. [89] selects station-keeping points with low sensitivity to navigation and execution errors, then fits unmodeled accelerations through reinforcement learning. The learned correction improves prediction in subsequent cycles while retaining the orbital model. Ref. [90] updates value and tracking-policy approximations online under unknown nonlinearities and partial observability. Both approaches use operational data to improve adaptation, with data coverage and identification accuracy limiting the reliability of the updated model or policy.

### *5.4. Fault-Aware Guidance and Control Reconfiguration*

Models of disturbances and parameter deviations usually assume that input and observation channels remain available. Actuator failures and sensor faults, by contrast, can directly change the available control capability. Ref. [30] distinguishes passive and active schemes. Passive fault tolerance is designed in advance to cover a specified fault set. It responds quickly, but may sacrifice nominal performance. Active fault tolerance relies on fault detection, isolation, or estimation and reconfigures the control after fault information becomes available. It offers more flexible coverage, but incurs diagnosis delay and the cost of misclassification.

After a fault, the remaining thrust, torque, sensing accuracy, and maneuver time limit the reachable terminal conditions. Replanning can retain the original target if sufficient control authority remains; otherwise, the target or mission mode must change. Diagnosis supplies fault information, capability assessment defines the revised feasible set, and guidance selects an executable trajectory [30]. The missed-thrust study in Section 6.3 provides quantitative mission-completion evidence for temporary thrust interruptions. Its success rates appear separately from nominal adaptive tracking in Table 1. They do not measure fault-detection latency, physical actuator repair, or recovery from permanent thrust loss.

Uncertainty-aware guidance combines prior information with observations acquired during flight. Robust formulations use bounded errors to maintain constraint margins; stochastic formulations allocate risk through statistical models; adaptive estimation reduces predictable model mismatch. Fault-tolerant guidance extends these functions by updating the admissible control set and, when necessary, the mission objective. Their joint design balances conservatism, information quality, control effort, and post-fault feasibility.

## 6. Learning-Enhanced Intelligent Guidance Methods

Learning can reuse information from optimized trajectories, dynamics propagation, or simulated interactions to reduce repeated computation [16,32]. Supervised methods learn mappings from solutions of the constrained problem in Section 2. Learned initial guesses or surrogates assist the numerical solution, while reinforcement learning defines a sequential objective through rewards. The relationship between that reward and Equations (9)–(11) must be stated for an optimal-control interpretation. Figure 5 distinguishes these roles and their execution paths.

### *6.1. Learning the Optimal Control Mapping*

Supervised learning approximates the control mapping represented by optimized trajectories. In ref. [32], a mass-optimal reaction-wheel landing policy succeeded in 98.3% of 1000 tests within the studied initialization region. Terminal tolerances were 10 m, 0.7 m/s, and 1°. Successful trajectories had a mean relative cost error of 0.72% against numerical references stopped at the same target distance. Extending the initial x and z ranges by 200 and 2000 m in set A2 reduced success to 34.9%. A separate benchmark used a network with four hidden layers and 32 units per layer. It averaged 0.092 ms over 100,000 inferences in Theano on an Intel Xeon E5-2687 W at 3.10 GHz. The success rates describe the tested landing cases; the timing measures network inference only. Neither quantity establishes full-update timing or performance outside the represented state range.

Higher-fidelity training trajectories can represent dynamics absent from a point-mass model. Ref. [91] generated hypersonic reentry trajectories with translational and rotational dynamics and three-axis control torques, then trained a network to predict feedback actions. The online stage evaluates that learned mapping after the trajectory optimization has been performed offline. Tests with inertia variations showed increasing deviation from the

reference optimal solution as uncertainty grew. Extending coverage requires additional high-quality trajectories and increases the offline optimization and training cost.

The training trajectories define the states, active constraints, and model behavior available to a supervised policy. New faults or unfamiliar trajectory structures can move the vehicle beyond that coverage. Direct policy evaluation is consequently useful for recurring tasks within a characterized operating region. Learned initialization or model approximation can cover a different role, reducing computation while retaining numerical correction.

### *6.2. Learning-Enhanced Numerical Optimization*

Repeated dynamics propagation is a major cost in trajectory search. Multiple-gravity-assist design, for example, evaluates many candidates through three-body integration. Ref. [92] learns a mapping between orbital parameters before and after a flyby from numerical propagation data. Ref. [93] embeds the surrogate in multiple-gravity-assist optimization and extends its inputs, accelerating candidate evaluation within the original search framework. The optimization retains its mission objective and constraints, while surrogate accuracy influences the final trajectory, particularly over multiple flyby segments. High-fidelity propagation remains the reference for assessing accumulated approximation error.

Learning can reduce the cost of initializing indirect optimal-control solvers. In ref. [94], networks predict initial costates for collision-free asteroid landing. The predictions either generate approximate trajectories directly or initialize an iterative solver. For 2000 trajectories at each asteroid, the reported mean costate-solution times using a DNN-based guess are 8.911 s for Bennu and 60.64 s for Eros, compared with 38.43 and 136.20 s for the baseline labeled Random. Network prediction alone takes 0.00907 s. These measurements use an Intel Core i9-7920X at 2.90 GHz; the warm-start measurements include iterative costate solution and must not be replaced by prediction time. Directly predicted trajectories require reserved collision-clearance margins, and their reported success changes with those margins.

Ref. [95] combines smooth penalties and double homotopy for time-optimal asteroid landing. One network predicts the minimum glide-slope constraint angle; another predicts initial costates and landing time. On an Intel Core i9-7920X at 2.90 GHz, prediction takes 0.009 s, compared with approximately 5.46 s for the double-homotopy method. Direct propagation from predicted values in 2000 sampled cases gives mean terminal position and velocity errors of 1 m and 0.025 m/s, with standard deviations of 0.9 m and 0.014 m/s. Predictions can also initialize double homotopy for higher accuracy, but a separate end-to-end warm-start runtime is not reported. The timing comparison therefore concerns prediction versus numerical solution at different accuracy levels, not a measured speedup of the refined solver.

Learning-enhanced optimization approximates expensive operations while retaining a trajectory objective and constraints. Surrogates replace propagation, and predicted costates or controls supply initial guesses. The examples above distinguish direct prediction from iterative refinement. Table 1 reports their timing scopes and accuracy separately from policy inference. A computational benefit must be judged at the accuracy and feasibility required by the mission.

### *6.3. Reinforcement Learning and Hybrid Guidance*

Reinforcement learning (RL) trains a feedback policy through interactions with the environment, using rewards instead of optimal-action labels. Aerospace applications include orbital transfers, attitude control, rendezvous and docking, and mission planning [16]. Actor-critic methods provide continuous control actions suitable for thrust and attitude commands [96]. Training shifts substantial computation offline, enabling rapid policy

evaluation during execution. The reward, dynamics model, and exploration strategy jointly determine the learned behavior and its relationship to the original performance objective.

Terminal-only rewards provide little information during a long low-thrust transfer. Ref. [97] supplies time-aligned nominal states and dense rewards, and trains policies for temporary thrust interruptions. For its 40-step, 348.79-day Earth-Mars transfer, 1000 Monte Carlo trials per scenario give rendezvous success rates of 82.5% for one missed-thrust step and 73.5% for interruptions lasting up to three consecutive steps. The latter model assigns a 0.9 recovery probability at each subsequent step. Success requires terminal position and velocity errors, normalized by the respective target magnitudes, to remain within $10^{-3}$. The corresponding mean final masses are 586.97 and 577.33 kg from an initial 1000 kg. Late interruptions can exceed the remaining correction capability, producing failed rendezvous and large terminal-error tails. The results support tolerance to the modeled temporary outages, not fault diagnosis or permanent actuator loss.

Gravity-assist transfers must leave subsequent approach phases reachable. Ref. [98] includes an analytical reachability indicator in the transfer reward and trains separate approach policies. For approach, reinforcement learning selects a zero-effort-miss/zero-effort-velocity (ZEM/ZEV) gain and an eigenvalue-related parameter; the remaining gain follows analytically. The robust ZEM/ZEV (R-ZEM/ZEV) law thus combines structured feedback with learned parameter selection. Its performance deteriorates under larger disturbances. Successful correction requires both an adequate reachability estimate and sufficient control authority within the conditions covered by training.

Hybrid guidance assigns learning and feedback control to different tasks. In ref. [4], a learned policy generates velocity commands that a conventional controller tracks. Tests on an air-bearing platform demonstrate this arrangement for spacecraft proximity operations. In ref. [89], learning estimates unmodeled acceleration within an orbital predictor-corrector. Both retain physical models or feedback around the learned component. Training coverage, tracking dynamics, and the available observations limit the conditions in which each arrangement can operate.

### *6.4. Safety Constraints and Engineering Validation*

Safety-constrained learning addresses exploration during training, constraint satisfaction during execution, and verifiable closed-loop behavior. The survey in ref. [99] relates policy constraints to sampling requirements and evaluation conditions. Conditional value-at-risk constraints provide one way to limit high-loss tail events in a Markov decision process [100]. For flight guidance, probabilistic risk measures must be related to physical state and input constraints, including those requiring continuous satisfaction. This motivates combining policy learning with constraint enforcement and closed-loop verification.

RTA monitors learned commands and applies corrective control when required. Ref. [17] compared six training configurations on the same 10 satellite-docking initial conditions. With RTA disabled during testing, speed-constraint violations occupied 92.0% of time steps after unprotected training and 37.7% after high-penalty training. Switching-RTA training reduced this fraction to 0.0%; the optimization-based cases gave 0.6% and 0.2%. All 10 docking cases succeeded, although their constraint-violation rates differed. Enabling RTA during testing maintained the studied constraint through interventions. Average training durations were 4.55 h without RTA, 5.13 h with explicit switching, and 10.59 h with implicit optimization. These offline runs used 20 processors on a Dell Precision Tower 5810. They illustrate the effect of training and corrective control on the tested constraint, without establishing online intervention latency.

Formal verification analyzes the closed loop formed by a learned policy and the vehicle dynamics over a prescribed state domain. For two-dimensional docking, ref. [18] incorporates verifiability into policy training and compares inductive proofs, neural Lyapunov barrier certificates, and reachability abstractions. Spurious paths and computational complexity prevented the tested reachability abstraction from establishing the required guarantee. The result highlights the difficulty of scaling verification while retaining sufficient dynamic detail. Explainability methods [101] complement this analysis by helping interpret network behavior, whereas constraint guarantees require a verified relationship among the policy, dynamics, and admissible states.

Learning can approximate control actions, accelerate selected solver operations, or optimize sequential decisions. Direct policies offer rapid evaluation within their training coverage. Learned initialization retains numerical refinement, whereas reinforcement learning handles delayed rewards and multistage choices. Architecture selection must account for offline data cost and the online correction still needed. RTA and verification connect learned commands to explicitly modeled physical constraints.

## 7. System-Level Autonomous Guidance Methods

System-level guidance connects trajectory decisions to control execution, other vehicles, and later mission phases. The coupling terms in Equations (1) and (4) represent interactions that may affect feasibility. Terminal conditions in Equation (6) also serve as handoff conditions between phases. Integrated control, coordination, and planning provide the interfaces through which these decisions are implemented [6,102]. This section distinguishes methods that optimize an explicit objective from supporting control or scheduling mechanisms. Such mechanisms are included to explain execution and information constraints, not as evidence that every system-level architecture solves an optimal control problem. Figure 6 summarizes these interfaces.

### *7.1. Coupled Design of Guidance and Control*

Separate guidance and control facilitates modular design and verification. However, acceleration, attitude, or angular-rate commands generated by the guidance layer must pass through inner-loop dynamics and actuators before they become actual vehicle motion. Short terminal times, aggressive maneuvers, or strong translation-rotation coupling can invalidate an assumption of instantaneous, accurate command tracking. Trajectory constraints may then fail during execution. IGC therefore includes guidance errors and vehicle dynamics within a common design. This reduces successive command conversions, but also increases model dimension, dynamic coupling, and verification burden [102].

Partial integration retains separate loops when the dynamics admit a useful time-scale separation. The outer loop incorporates guidance and vehicle-motion information, while the inner loop tracks attitude or angular-rate commands. Ref. [103] uses MPSP or model predictive spread control (MPSC) in the outer loop and dynamic inversion in the inner loop. The design reduces miss distance relative to a single-loop scheme in six-degree-of-freedom simulations. Ref. [104] similarly connects relative-position errors to angular-rate and thrust commands through two loops. These architectures combine dynamic coupling where it affects guidance with hierarchical control where fast tracking remains effective.

Robust integrated guidance and control addresses uncertain parameters within the coupled dynamics. Ref. [35] tested 1000 missile-interception cases with mass variations up to 10 kg, pitch-inertia variations up to 10 kg·m$^2$, and aerodynamic-coefficient variations up to ±30%. All reported miss distances were below 0.35 m, with interception times of 5.0905–5.0935 s. Processor and computation time were not reported. These results characterize accuracy under the tested perturbations; they do not establish an optimality

gap or computation deadline. Ref. [105] identifies interception angle, actuator saturation, and control energy as further design considerations. Integration must account for these requirements as well as tracking performance.

Spacecraft position-attitude coupling depends on mission time scales and actuator configuration. Proximity operations can couple translational and rotational motion more strongly than orbital maneuvers with a clear time-scale separation. Thrust misalignment and environmental forces further affect this relationship. The review in ref. [21] organizes integrated position-attitude models and controllers according to these conditions. Joint planning is most valuable for states whose interactions materially affect accuracy or feasibility, while other dynamics can retain a hierarchical implementation.

### *7.2. Multi-Vehicle Coordination and Distributed Learning*

Multi-vehicle missions couple individual trajectories through safety, communication, and shared objectives. A maneuver that improves one vehicle's motion may reduce separation or disrupt a communication link. Relative-motion control under six-degree-of-freedom dynamics [104] therefore addresses only part of spacecraft-swarm reconfiguration. Cooperative guidance additionally accounts for target formation, connectivity, collision avoidance, and uncertainty. The control problem combines vehicle dynamics with the evolution of the information network.

Learning-based coordination can combine cost-related decisions with established feedback structures. Ref. [2] integrates actor-critic learning, artificial potential fields, and backstepping for collision avoidance. Four spacecraft and a virtual leader use point-mass relative dynamics, a fully connected balanced graph, bounded disturbances, and a moving obstacle. Simulations run in MATLAB/Simulink on a four-core Intel i7 at 3.20 GHz with 7.88 GB RAM. Computation is described as approximately comparable to backstepping with potential fields, without numerical runtime or statistical success rates. This supports a coordination example under the stated network assumptions. It does not establish a common optimality measure or a runtime advantage over the other methods in Table 1.

Ref. [106] uses dimension-matching matrices for three-dimensional unmanned aerial vehicle (UAV) leaders and two-dimensional ground followers. Distributed observers and fixed-time protocols achieve formation and containment under the stated disturbance and communication assumptions. A well-informed follower is linked to all formation leaders. The settling-time bound is independent of the initial state and is set by protocol parameters. This is a supporting coordination result rather than an optimal-control benchmark. It is included because heterogeneous dynamics and information exchange constrain the trajectories that a distributed optimizer can implement.

Distributed guidance couples local control with network-wide feasibility. Communication topology determines which relative states are available, while actuator limits and collision constraints restrict coordinated maneuvers. Effective design therefore links convergence of the coordination law to minimum separation, communication load, and achievable vehicle motion.

### *7.3. Coordination Between Planning and Execution*

Mission planning connects discrete route selection with continuous flight dynamics. Ref. [107] represents routes around no-fly zones as graph paths and evaluates candidates through path-following guidance and dynamic integration. A graph attention network then accelerates the upper-level search and supplies an initial route for continuous trajectory optimization. Dynamic refinement relates the selected graph path to physical path constraints and control cost. Reduced prediction accuracy on graph sizes far from the training

set highlights the importance of structural coverage when transferring learned routing to new missions.

Ref. [108] treats long-term regional coverage and coverage balance as an integer scheduling problem, solved by a two-stage hybrid method. Its constraints include exclusive task execution, memory, energy, attitude-maneuvering time, solar elevation, and cloud cover. This is mission-level scheduling rather than continuous-time optimal guidance. It is included to show how task assignments and resource windows constrain the trajectories and attitude maneuvers required for execution.

Learning also links continuous trajectory design to execution. Ref. [91] optimizes coupled translational and rotational trajectories offline and learns a state-to-control mapping; inertia variations expose limits in training coverage. This role differs from graph-based routing and discrete resource scheduling. The pseudospectral convex planning example discussed in Section 4.3 belongs to continuous trajectory optimization, rather than establishing a general system-autonomy capability. Each planning level must pass feasible commands or constraints to the level that executes them.

### *7.4. Multistage Missions and Fault-Driven Replanning*

Multistage guidance requires each phase to end within the feasible starting conditions of the next. In ref. [98], an analytical reachability indicator shapes the transfer reward, and a separately trained approach policy selects ZEM/ZEV parameters. The indicator uses a short arc and an empirical relation between available velocity increment and low-thrust approach capability. It encourages transfer endpoints that retain correction authority. Performance decreases under larger disturbances, and navigation is outside the model. The example therefore does not establish feasibility under combined estimation and guidance errors.

An engine failure changes the resources available for the remaining flight. Ref. [61] considers an engine-out aircraft approaching a specified target region with wind preview. Long-horizon economic MPC maximizes arrival altitude, while shorter-horizon MPC tracks the reference. Favorable wind improves the predicted glide, but preview error and sensor noise affect the result. This is replanning for a prescribed degraded condition. It does not quantify the time required to detect the failure or select an alternative landing site.

System-level guidance integrates functions according to the dominant mission couplings. Strong attitude or actuator effects favor joint guidance-control design. Collision avoidance and shared objectives require coordinated decisions across vehicles. Multistage missions require terminal conditions compatible with subsequent maneuvers, while faults can require changes in both trajectory and target. These couplings connect local control performance to mission completion and provide the basis for choosing the required level of integration.

## 8. Conclusions and Outlook

Optimal guidance has developed from offline trajectory computation toward closed-loop, uncertainty-aware, and increasingly autonomous mission execution. Classical optimization provides the numerical foundation; real-time methods embed it in feedback; robust, stochastic, and adaptive methods address model uncertainty; learning accelerates selected computations and decision processes. Integrated and distributed designs extend these capabilities to coupled vehicles and multistage missions. The comparison below builds on the control-capability perspective [19] and the eight dimensions introduced in Section 2.

### *8.1. Comparison of Methods and Summary of Capabilities*

Table 1 presents study-specific evidence across the eight dimensions defined in Section 2. S denotes direct evidence for at least one stated indicator, P indirect or incomplete evidence, and N a dimension not addressed by the example. Separate profiles distinguish methods that share a row. Additional numerical examples cover global search, lossless convexification, stochastic covariance control, missed-thrust tolerance, and both learned-costate studies. Section pointers identify their existing source discussions. IGC and coordination retain separate profiles and metrics. The entries show evidence coverage and within-study trade-offs, not rankings across missions, processors, or accuracy requirements. Unreported detection, recovery, and complete-update timings remain NR (not reported).

**Table 1.** Representative evidence across eight evaluation dimensions. Profile order: optimality, adaptability, robustness, intelligence, fault tolerance, real-time capability, distributed coordination, and functional integration. S = supported for a stated indicator; P = partial evidence; N = not addressed by the cited example. NR = not reported; IGC = integrated guidance and control; RTA = runtime assurance. Profiles describe evidence coverage within the reported conditions, not universal method capabilities.

| Method Route | Representative Sources | Eight-Dimension Profile and Representative Results | Applicability and Test Conditions |
| --- | --- | --- | --- |
| Classical optimization and global search | [26]; global search: Section 3.4 | Gauss-collocation profile: P/N/N/N/N/N/N/N. At 40 points, maximum nodal state/costate/control errors are $9 \times 10^{-7}$/$1.23 \times 10^{-5}$/$4.2 \times 10^{-6}$. Global-search profile: P/N/N/N/N/P/N/N. Hybrid particle-swarm/gravitational search, standard particle swarm, and gravitational search give mean objectives of $9.37/9.45/14.16 \times 10^{-4}$ and mean runtimes of 66.64/102.83/66.94 s. | Collocation uses a scalar analytical reference and TOMLAB/SNOPT; runtime and processor NR. Global search uses 100 runs, three design variables, MATLAB 2016, and Core i3 at 2.53 GHz. Runtime standard deviations are 13.90/9.14/18.08 s. Its objective is problem-specific; these data do not certify global optimality. |
| Receding-horizon optimization and fast local correction | [59,60,65,66]; quantitative example [65]. | Profile: P/S/N/N/N/P/N/N. MPC updates decisions through feedback; the GMPSP example quantifies local correction. Eight iterations reduce terminal-angle errors from 0.42° and 2.5° to below 0.01°, while range error increases from 0.2 to 0.5 m. | The example includes first-order autopilot delay and a 50 ms simulation step. Processor and wall-clock solve time NR. The step size does not demonstrate deadline compliance. Angle accuracy improves at the cost of increased range error in this case. |
| Lossless convexification and sequential convex programming | [75]; lossless convexification: Section 4.3 | Lossless-convexification profile: P/N/N/N/N/N/N/N. Fuel use is 387.9/390.4/399.5 kg and flight time 72/75/81 s without the no-subsurface constraint, with it, and with an added glide slope. SCP profile: P/N/N/N/N/P/N/N. Fourth-order soft/hard trust regions give 100%/42% convergence, 17.11/15.50 ms reported CPU time, and mean objectives 0.26073/0.26065. | The landing example uses SeDuMi/MATLAB; the first trajectory enters the subsurface. Fuel differences reflect constraint changes; solver runtime and processor NR. SCP uses 100 random references, 100 nodes, ECOS, and Ryzen 7 5800H at 3.20 GHz. Its timing scope and treatment of failed runs remain unspecified. |

**Table 1.** *Cont.*

| Method Route | Representative Sources | Eight-Dimension Profile and Representative Results | Applicability and Test Conditions |
| --- | --- | --- | --- |
| Robust and stochastic optimization | [29]; stochastic example: Section 5.2 | Tube-MPC profile: P/N/S/N/N/N/N/N. Three experiments preserve keep-out constraints with 12%/13%/17% more velocity increment. Stochastic-covariance profile: P/N/S/N/N/P/N/N. Open/closed-loop objectives are 8.898/13.640 km/s; predicted terminal x-position variances are $2.710 \times 10^{14}$/$1.000 \times 10^{11}$ km$^2$. Monte Carlo variances are $2.910 \times 10^{14}$/$9.357 \times 10^{10}$ km$^2$. | Tube MPC uses a planar air-bearing platform and matched dynamics/weights; processor and runtime NR. Stochastic control uses an Earth-Mars transfer, C++/MOSEK, and 41 nodes; 10 iterations take about 5.8 s, processor NR. The sampled terminal y variance is $1.074 \times 10^{11}$ km$^2$ versus a $1.000 \times 10^{11}$ km$^2$ target. The closed-loop Monte Carlo 95th-percentile effort is 11.388 km/s; an empirical chance-constraint violation rate is not reported. |
| Online estimation and fault tolerance | [27]; missed-thrust example: Sections 5.4 and 6.3 | Adaptive-tracking profile: N/S/P/P/N/P/N/P. Mean x/y/z errors are 0.3628/0.6530/0.0008 m versus 0.4454/0.8427/1.9378 m for H-infinity control. Missed-thrust profile: P/N/P/S/S/N/N/N. Earth-Mars rendezvous success is 82.5% for a single missed-thrust step and 73.5% for up to three consecutive missed steps. Mean final masses are 586.97/577.33 kg. | Tracking uses Raspberry Pi 3/X-Plane HIL and 6.8 m/s wind; maximum x-error increases from 4.0226 to 4.8695 m. Outage results use a 40-step, 348.79-day transfer, 1000 trials per scenario, 1000 kg initial mass, and $10^{-3}$ relative terminal position/velocity tolerances. Modeled recovery probability is 0.9 per subsequent step. Detection/physical recovery times NR; late outages can cause failure. |
| Learning mappings and learning-enhanced solvers | [32]; learned costates: Section 6.2 | Policy profile: P/N/P/S/N/P/N/N. Landing success is 98.3%, mean cost error 0.72%; separate inference benchmark 0.092 ms. Learned-costate profiles: P/N/N/S/N/P/N/N. Zhao et al. report iterative warm-start times of 8.911/60.64 s for Bennu/Eros versus 38.43/136.20 s baseline. Hu et al. report 0.009 s prediction versus 5.46 s double homotopy, with mean direct-prediction terminal errors of 1 m and 0.025 m/s. | Policy tests use 1000 cases and 10 m/0.7 m/s/1° tolerances; success falls to 34.9% in A2. Its inference benchmark uses Xeon E5-2687 W at 3.10 GHz. Both costate studies use Core i9-7920X at 2.90 GHz. Zhao et al. use 2000 trajectories per asteroid; Hu et al. use 2000 direct-prediction cases and do not separately time warm-start refinement. The timing scopes and accuracies differ. |
| Reinforcement learning and safety assurance | [17] | Profile: N/N/N/S/N/N/N/S. Without test-time RTA, docking speed-constraint violations occupy 92.0% of steps after unprotected training versus 0.0% after switching-RTA training. All 10 cases succeed, so task success alone does not characterize constraint satisfaction. | The evidence concerns learning with a corrective safety function. Offline training takes 4.55 h unprotected and 5.13 h with explicit switching on a Dell 5810 with 20 processors. Monitor and intervention latency NR; training duration is not an online timing metric. |

**Table 1.** *Cont.*

| Method Route | Representative Sources | Eight-Dimension Profile and Representative Results | Applicability and Test Conditions |
| --- | --- | --- | --- |
| IGC; distributed coordination (separate examples) | [2,35] | IGC profile: N/N/S/N/N/N/N/P. Miss below 0.35 m in 1000 perturbed missile simulations; an integration benefit is not isolated by this metric. Coordination profile: P/N/P/P/N/N/S/N. Four spacecraft and a virtual leader; numerical runtime and statistical success rate NR. The profiles and metrics are separate. | IGC tests mass, inertia, and aerodynamic perturbations. The swarm uses point-mass dynamics, a fully connected graph, MATLAB/Simulink, a four-core i7 at 3.20 GHz, and 7.88 GB RAM. Interception accuracy and coordinated motion are not compared through a common performance score. |

Method selection starts with the role of optimization in the mission. Offline design prioritizes trajectory accuracy and constraint resolution. Indirect methods exploit tractable optimality structure, while direct transcription accommodates complex models. Online guidance also requires repeated state correction within a computation deadline. MPC organizes this feedback; local solvers and convexification supply candidate trajectories. Lossless reformulation offers guarantees under its structural assumptions, whereas sequential approximation treats broader nonlinear models. Table 1 separates numerical accuracy and convergence evidence from measurements needed to establish complete guidance-update timing.

Uncertainty and mission coupling determine which additional functions are needed. Robust methods protect against bounded errors, stochastic methods allocate risk, and adaptation updates models from observations. Learning can reduce repeated computation within a characterized training range. Integrated and distributed designs address execution dynamics, inter-vehicle constraints, and mission interfaces. The reported examples illustrate conditional trade-offs. In the single SCP comparison, the soft trust region converged more often and had a higher reported mean CPU time, with an unresolved timing scope. The tube-MPC experiments used additional control effort to preserve keep-out margins. The learned landing policy lost success rate when tested outside its original initialization range. These observations motivate mission-specific comparisons; they do not establish universal advantages, embedded deadline compliance, or flight heritage.

### *8.2. Research Outlook*

Further progress depends on coordinating optimization, uncertainty treatment, learning, and execution within a common mission design. The comparison identifies four related problems: coupled-dynamics planning, computation-constrained guidance, adaptation to uncertainty and degraded capability, and feasibility across vehicles and mission phases. Powered descent, spacecraft rendezvous, and formation reconfiguration provide representative settings for investigating these problems.

Coupled-dynamics planning requires a principled choice of model fidelity. In powered descent, thrust misalignment, attitude motion, and actuator lag can alter both propellant demand and terminal accuracy. Retaining every dynamic state increases the optimization burden, while excessive reduction can produce trajectories that are difficult to track. Adaptive-fidelity models and hierarchical optimization offer promising directions. Their development requires understanding how neglected dynamics affect continuous path constraints and how planning accuracy translates into closed-loop performance under a common six-degree-of-freedom model.

Computation-constrained guidance must handle interrupted optimization or failed convergence. Warm-start degradation, abrupt navigation updates, and processor limits can prevent a new trajectory from meeting its deadline. Reusing a feasible plan, applying local correction, or invoking backup control offers possible responses. Each remains valid only for particular states and interruption durations. Termination rules and recovery control should be designed together. Powered descent offers a setting for examining update latency, constraint satisfaction, terminal accuracy, and propellant use on the intended hardware.

Adaptive uncertainty and fault handling require reliable decisions from incomplete observations. Navigation bias, changing disturbance statistics, and loss of thruster effectiveness can occur together during rendezvous and docking. Estimation-based or learned updates can reduce prediction error, while residual uncertainty still determines the required control margin. A central question is when improved information permits less conservative guidance and when capability loss requires a different target. Joint estimation, diagnosis, and replanning should relate observation quality and diagnosis delay to achievable docking accuracy, collision risk, and fuel consumption.

Cooperative multistage autonomy requires local decisions to preserve both group safety and later maneuver capability. Formation reconfiguration under communication delay, link loss, and heterogeneous actuator limits is a representative challenge. The terminal formation must remain compatible with the resources and geometry required by the next maneuver. Distributed optimization, reachability analysis, and mission replanning offer complementary tools, but their information exchange and computation requirements need coordinated design. Consistent simulation and hardware-in-the-loop studies can clarify the trade-offs among minimum separation, communication load, fuel expenditure, and completion of successive mission phases.

**Author Contributions:** Conceptualization, D.F.; Formal analysis, R.H.; Funding acquisition, Y.N.; Investigation, G.Y.; Methodology, D.F.; Resources, B.W.; Supervision, R.H.; Validation, R.H.; Writing—original draft, D.F.; Writing—review and editing, R.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** Supported by the State Key Laboratory of High-Efficiency Reusable Aerospace Transportation Technology (No. RATT-KF03-2025011).

**Data Availability Statement:** No new data were created or analyzed in this study. Data sharing is not applicable to this article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Xie, L.; Zhou, X.; Zhang, H.-B.; Tang, G.-J. An improved Convex Optimization-Based Guidance for Fuel-Optimal Powered Landing. *Adv. Space Res.* **2024**, *74*, 3256–3272. [CrossRef]
2. Sun, J.; Meng, Y.; Huang, J.; Liu, F.; Li, S. Distributed Cooperative Control with Collision Avoidance for Spacecraft Swarm Reconfiguration via Reinforcement Learning. *Acta Astronaut.* **2023**, *205*, 95–109. [CrossRef]
3. Yu, G.; Xie, L.; Sun, S.; Nan, Y. Stochastic Multi-Pass Noncoplanar Aeroassisted Orbital Transfer Problem using Log-Domain Sequential Covariance Control. *Aerosp. Sci. Technol.* **2027**, *180*, 113713. [CrossRef]
4. Hovell, K.; Ulrich, S. Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance. *J. Spacecr. Rockets* **2021**, *58*, 254–264. [CrossRef]
5. Xie, L.; Zhou, X.; Zhang, H.-B.; Tang, G.-J. Hybrid-Order Soft Trust Region-Based Sequential Convex Programming for Reentry Trajectory Optimization. *Adv. Space Res.* **2024**, *73*, 3195–3208. [CrossRef]
6. Chai, R.; Tsourdos, A.; Savvaris, A.; Chai, S.; Xia, Y.; Philip Chen, C.L. Review of Advanced Guidance and Control Algorithms for Space/Aerospace Vehicles. *Prog. Aerosp. Sci.* **2021**, *122*, 100696. [CrossRef]
7. Li, S.; Jiang, X. Review and Prospect of Guidance and Control for Mars Atmospheric Entry. *Prog. Aerosp. Sci.* **2014**, *69*, 40–57. [CrossRef]

8. Yang, H.; Hu, J.; Bai, X.; Li, S. Review of Trajectory Design and Optimization for Jovian System Exploration. *Space Sci. Technol.* **2023**, *3*, 36. [CrossRef]
9. Chai, R.; Savvaris, A.; Tsourdos, A.; Chai, S.; Xia, Y. A review of optimization techniques in spacecraft flight trajectory design. *Prog. Aerosp. Sci.* **2019**, *109*, 100543. [CrossRef]
10. Ding, Y.; Yue, X.; Chen, G.; Si, J. Review of Control and Guidance Technology on Hypersonic Vehicle. *Chin. J. Aeronaut.* **2022**, *35*, 1–18. [CrossRef]
11. Betts, J.T. Survey of Numerical Methods for Trajectory Optimization. *J. Guid. Control Dyn.* **1998**, *21*, 193–207. [CrossRef]
12. Ross, I.M.; Karpenko, M. A review of Pseudospectral Optimal Control: From Theory to Flight. *Annu. Rev. Control* **2012**, *36*, 182–197. [CrossRef]
13. Izzo, D.; Märtens, M.; Pan, B. A Survey on Artificial Intelligence Trends in Spacecraft Guidance Dynamics and Control. *Astrodynamics* **2019**, *3*, 287–299. [CrossRef]
14. Eren, U.; Prach, A.; Koçer, B.B.; Raković, S.V.; Kayacan, E.; Açıkmeşe, B. Model Predictive Control in Aerospace Systems: Current State and Opportunities. *J. Guid. Control Dyn.* **2017**, *40*, 1541–1566. [CrossRef]
15. Malyuta, D.; Yu, Y.; Elango, P.; Açıkmeşe, B. Advances in Trajectory Optimization for Space Vehicle Control. *Annu. Rev. Control* **2021**, *52*, 282–315. [CrossRef]
16. Tipaldi, M.; Iervolino, R.; Massenio, P.R. Reinforcement Learning in Spacecraft Control Applications: Advances, Prospects, and Challenges. *Annu. Rev. Control* **2022**, *54*, 1–23. [CrossRef]
17. Dunlap, K.; Mote, M.; Delsing, K.; Hobbs, K.L. Run Time Assured Reinforcement Learning for Safe Satellite Docking. *J. Aerosp. Inf. Syst.* **2023**, *20*, 25–36. [CrossRef]
18. Mandal, U.; Amir, G.; Wu, H.; Daukantas, I.; Newell, F.L.; Ravaioli, U.; Meng, B.; Durling, M.; Hobbs, K.; Ganai, M.; et al. Safe and Reliable Training of Learning-Based Aerospace Controllers. In *Proceedings of the 2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*; IEEE: San Diego, CA, USA, 2024. [CrossRef]
19. Nan, Y.; Chen, H.; Yang, Y.; Lv, K. Primary Methodologies of Modern Control: Status and Prospect. *J. Nanjing Univ. Aeronaut. Astronaut.* **2015**, *47*, 798–810. [CrossRef]
20. Sedelinkov, A.; Nikolaeva, A.; Serdakova, V.; Khnyryova, E. Technologies for Increasing the Control Efficiency of Small Spacecraft with Solar Panels by Taking into Account Temperature Shock. *Technologies* **2024**, *12*, 207. [CrossRef]
21. Zhang, F.; Duan, G. Coupled Dynamics and Integrated Control for Position and Attitude Motions of Spacecraft: A Survey. *IEEE/CAA J. Autom. Sin.* **2023**, *10*, 2187–2208. [CrossRef]
22. Prabhakar, A.; Fisher, J.; Bhattacharya, R. Polynomial Chaos-Based Analysis of Probabilistic Uncertainty in Hypersonic Flight Dynamics. *J. Guid. Control Dyn.* **2010**, *33*, 222–234. [CrossRef]
23. Jones, B.A.; Doostan, A.; Born, G.H. Nonlinear Propagation of Orbit Uncertainty Using Non-Intrusive Polynomial Chaos. *J. Guid. Control Dyn.* **2013**, *36*, 430–444. [CrossRef]
24. Xu, Y.; Liu, P.; Yang, B.; Cao, L.; Li, S. Parameter Estimation via Multi-Task GRU for Incomplete-Information Orbital Pursuit-evasion Games. *Aerosp. Sci. Technol.* **2026**, *176*, 112093. [CrossRef]
25. Geller, D.K. Linear Covariance Techniques for Orbital Rendezvous Analysis and Autonomous Onboard Mission Planning. *J. Guid. Control Dyn.* **2006**, *29*, 1404–1414. [CrossRef]
26. Benson, D.A.; Huntington, G.T.; Thorvaldsen, T.P.; Rao, A.V. Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method. *J. Guid. Control Dyn.* **2006**, *29*, 1435–1440. [CrossRef]
27. Lai, Y.-C.; Le, T.-Q. Adaptive Learning-Based Observer with Dynamic Inversion for the Autonomous Flight of an Unmanned Helicopter. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 1803–1814. [CrossRef]
28. Rawlings, J.B.; Mayne, D.Q.; Diehl, M. *Model Predictive Control: Theory, Computation, and Design*, 2nd ed.; Nob Hill Publishing: San Francisco, CA, USA, 2017.
29. Frekhaug, T.A.; Hudson, J.; Sanjurjo-Rivo, M.; Soler, M.; Romano, M. Robust Model Predictive Control for Proximity Operations with Experimental Demonstrations. *J. Guid. Control Dyn.* **2024**, *47*, 1688–1697. [CrossRef]
30. Hasan, M.N.; Haris, M.; Qin, S. Fault-Tolerant Spacecraft Attitude Control: A Critical Assessment. *Prog. Aerosp. Sci.* **2022**, *130*, 100806. [CrossRef]
31. Takagi, T.; Sugeno, M. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Trans. Syst. Man Cybern.* **1985**, *SMC-15*, 116–132. [CrossRef]
32. Sánchez-Sánchez, C.; Izzo, D. Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems. *J. Guid. Control Dyn.* **2018**, *41*, 1122–1135. [CrossRef]
33. Hong, H.; Maity, A.; Holzapfel, F.; Tang, S. Model Predictive Convex Programming for Constrained Vehicle Guidance. *IEEE Trans. Aerosp. Electron. Syst.* **2019**, *55*, 2487–2500. [CrossRef]
34. Olfati-Saber, R.; Fax, J.A.; Murray, R.M. Consensus and Cooperation in Networked Multi-Agent Systems. *Proc. IEEE* **2007**, *95*, 215–233. [CrossRef]

35. Wang, Q.; Ran, M.; Dong, C. Robust Partial Integrated Guidance and Control for Missiles via Extended State Observer. *ISA Trans.* **2016**, *65*, 27–36. [CrossRef] [PubMed]
36. Cherry, G. A General, Explicit, Optimizing Guidance Law for Rocket-Propelled Spaceflight. In *Proceedings of the Astrodynamics Guidance and Control Conference*; American Institute of Aeronautics and Astronautics: Los Angeles, CA, USA, 1964. [CrossRef]
37. Morrison, D.D.; Riley, J.D.; Zancanaro, J.F. Multiple Shooting Method for Two-Point Boundary Value Problems. *Commun. ACM* **1962**, *5*, 613–614. [CrossRef]
38. Trent, A.; Venkataraman, R.; Doman, D. Trajectory Generation Using a Modified Simple Shooting Method. In *Proceedings of the 2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*; IEEE: Big Sky, MT, USA, 2004; Volume 4, pp. 2723–2729. [CrossRef]
39. Jiang, F.; Baoyin, H.; Li, J. Practical Techniques for Low-Thrust Trajectory Optimization with Homotopic Approach. *J. Guid. Control Dyn.* **2012**, *35*, 245–258. [CrossRef]
40. Grant, M.J.; Braun, R.D. Rapid Indirect Trajectory Optimization for Conceptual Design of Hypersonic Missions. *J. Spacecr. Rockets* **2015**, *52*, 177–182. [CrossRef]
41. Pan, X.; Pan, B.; Li, Z. Bounding Homotopy Method for Minimum-Time Low-Thrust Transfer in the Circular Restricted Three-Body Problem. *J. Astronaut. Sci.* **2020**, *67*, 1220–1248. [CrossRef]
42. Pan, B.; Ni, Y.; Ma, Y.; Lu, P. Smoothing Homotopy Methods for Solving Nonlinear Optimal Control Problems. *J. Guid. Control Dyn.* **2023**, *46*, 1470–1484. [CrossRef]
43. Mehra, R.; Davis, R. A generalized Gradient Method for Optimal Control Problems with Inequality Constraints and Singular arcs. *IEEE Trans. Autom. Control* **1972**, *17*, 69–79. [CrossRef]
44. Mall, K.; Grant, M.J.; Taheri, E. Uniform Trigonometrization Method for Optimal Control Problems with Control and State Constraints. *J. Spacecr. Rockets* **2020**, *57*, 995–1007. [CrossRef]
45. Hargraves, C.R.; Paris, S.W. Direct Trajectory Optimization Using Nonlinear Programming and Collocation. *J. Guid. Control Dyn.* **1987**, *10*, 338–342. [CrossRef] [PubMed]
46. Broyden, C.G. Quasi-Newton Methods and Their Application to Function Minimisation. *Math. Comput.* **1967**, *21*, 368–381. [CrossRef] [PubMed]
47. Lasdon, L.; Mitter, S.; Waren, A. The Conjugate Gradient Method for Optimal Control Problems. *IEEE Trans. Autom. Control* **1967**, *12*, 132–138. [CrossRef]
48. Hestenes, M.R. Multiplier and Gradient Methods. *J. Optim. Theory Appl.* **1969**, *4*, 303–320. [CrossRef]
49. Fahroo, F.; Ross, I.M. Direct Trajectory Optimization by a Chebyshev Pseudospectral Method. *J. Guid. Control Dyn.* **2002**, *25*, 160–166. [CrossRef] [PubMed]
50. Garg, D.; Patterson, M.; Hager, W.W.; Rao, A.V.; Benson, D.A.; Huntington, G.T. A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods. *Automatica* **2010**, *46*, 1843–1851. [CrossRef]
51. Gong, Q.; Fahroo, F.; Ross, I.M. Spectral Algorithm for Pseudospectral Methods in Optimal Control. *J. Guid. Control Dyn.* **2008**, *31*, 460–471. [CrossRef]
52. Chai, R.; Tsourdos, A.; Savvaris, A.; Chai, S.; Xia, Y. High-Fidelity Trajectory Optimization for Aeroassisted Vehicles Using Variable Order Pseudospectral Method. *Chin. J. Aeronaut.* **2021**, *34*, 237–251. [CrossRef]
53. Yang, S.; Cui, T.; Hao, X.; Yu, D. Trajectory Optimization for a Ramjet-Powered Vehicle in Ascent Phase via the Gauss Pseudospectral Method. *Aerosp. Sci. Technol.* **2017**, *67*, 88–95. [CrossRef]
54. Murty, K.G.; Kabadi, S.N. Some NP-Complete Problems in Quadratic and Nonlinear Programming. *Math. Program.* **1987**, *39*, 117–129. [CrossRef]
55. Wu, Y. A Survey on Population-Based Meta-Heuristic Algorithms for Motion Planning of Aircraft. *Swarm Evol. Comput.* **2021**, *62*, 100844. [CrossRef]
56. Zhao, J.; Zhou, R. Particle Swarm Optimization Applied to Hypersonic Reentry Trajectories. *Chin. J. Aeronaut.* **2015**, *28*, 822–831. [CrossRef]
57. Shahzad Sana, K.; Hu, W. Hypersonic Reentry Trajectory Planning by Using Hybrid Fractional-Order Particle Swarm Optimization and Gravitational Search Algorithm. *Chin. J. Aeronaut.* **2021**, *34*, 50–67. [CrossRef]
58. Huang, G.; Lu, Y.; Nan, Y. A Survey of Numerical Algorithms for Trajectory Optimization of Flight Vehicles. *Sci. China Technol. Sci.* **2012**, *55*, 2538–2560. [CrossRef]
59. Mayne, D.Q.; Rawlings, J.B.; Rao, C.V.; Scokaert, P.O.M. Constrained Model Predictive Control: Stability and Optimality. *Automatica* **2000**, *36*, 789–814. [CrossRef]
60. Chai, R.; Savvaris, A.; Tsourdos, A.; Chai, S.; Xia, Y. Optimal Tracking Guidance for Aeroassisted Spacecraft Reconnaissance mission Based on Receding Horizon Control. *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 1575–1588. [CrossRef]
61. Fang, X.; Jiang, J.; Chen, W.-H. Model Predictive Control with Wind Preview for Aircraft Forced Landing. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *59*, 3995–4004. [CrossRef]

62. Leomanni, M.; Bianchini, G.; Garulli, A.; Quartullo, R. Sum-of-Norms Periodic Model Predictive Control for Space Rendezvous. *IEEE Trans. Contr. Syst. Technol.* **2022**, *30*, 1311–1318. [CrossRef]
63. Murray, D.M.; Yakowitz, S.J. Differential dynamic programming and Newton's method for discrete optimal control problems. *J. Optim. Theory Appl.* **1984**, *43*, 395–414. [CrossRef]
64. He, S.; Shin, H.-S.; Tsourdos, A. Computational Guidance Using Sparse Gauss-Hermite Quadrature Differential Dynamic Programming. *IFAC-Pap.* **2019**, *52*, 13–18. [CrossRef]
65. Maity, A.; Oza, H.B.; Padhi, R. Generalized Model Predictive Static Programming and Angle-Constrained Guidance of Air-to-Ground Missiles. *J. Guid. Control Dyn.* **2014**, *37*, 1897–1913. [CrossRef]
66. Mondal, S.; Padhi, R. Angle-Constrained Terminal Guidance Using Quasi-Spectral Model Predictive Static Programming. *J. Guid. Control Dyn.* **2018**, *41*, 779–787. [CrossRef]
67. Xu, X.; Chen, W.; Yang, L. Nonlinear Optimal Midcourse Guidance Using Sequential Pseudospectral Model Predictive Control. *IEEE Trans. Aerosp. Electron. Syst.* **2025**, *61*, 14463–14478. [CrossRef]
68. Malyuta, D.; Reynolds, T.P.; Szmuk, M.; Lew, T.; Bonalli, R.; Pavone, M.; Açıkmeşe, B. Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently. *IEEE Control Syst.* **2022**, *42*, 40–113. [CrossRef]
69. Acikmese, B.; Ploen, S.R. Convex Programming Approach to Powered Descent Guidance for Mars Landing. *J. Guid. Control Dyn.* **2007**, *30*, 1353–1366. [CrossRef]
70. Mao, Y.; Szmuk, M.; Açıkmeşe, B. Successive Convexification of Non-Convex Optimal Control Problems and Its Convergence Properties. In *Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC)*; IEEE: San Diego, CA, USA, 2016; pp. 3636–3641. [CrossRef]
71. Bonalli, R.; Cauligi, A.; Bylard, A.; Pavone, M. GuSTO: Guaranteed Sequential Trajectory Optimization via Sequential Convex Programming. In *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)*; IEEE: San Diego, CA, USA, 2019; pp. 6741–6747. [CrossRef]
72. Liu, X.; Li, S.; Xin, M. Mars Entry Trajectory Planning with Range Discretization and Successive Convexification. *J. Guid. Control Dyn.* **2022**, *45*, 755–763. [CrossRef]
73. Li, D.; Li, Y.; Liu, X.; Yang, B.; Huang, X.; Yang, Y.; Wang, B.; Li, S. Pseudospectral Convex Programming for Free-Floating Space Manipulator Path Planning. *Space Sci. Technol.* **2023**, *3*, 30. [CrossRef]
74. Liu, X.; Li, S.; Xin, M. Indirect Sequential Convex Programming for Mars Entry Terminal Altitude Maximization. *Acta Astronaut.* **2023**, *211*, 909–919. [CrossRef]
75. Xie, L.; Zhou, X.; Zhang, H.-B.; Tang, G.-J. Higher-Order Soft-Trust-Region-Based Sequential Convex Programming. *J. Guid. Control Dyn.* **2023**, *46*, 2199–2206. [CrossRef]
76. Xie, L.; Zhou, X.; Zhang, H.-B.; Tang, G.-J.; Li, S. High-Dimensional Merit-Function-Based Sequential Convex Programming for Nonlinear Trajectory Optimization. *J. Guid. Control Dyn.* **2025**, *48*, 1525–1544. [CrossRef]
77. Xie, L.; Gong, Z.; Zhang, H.; Tang, G.; Li, S. Penalty Separation-Based Sequential Convex Programming for Lunar Spacecraft Reentry Trajectory Optimization. *IEEE Trans. Aerosp. Electron. Syst.* **2026**, *62*, 646–660. [CrossRef]
78. Zhang, T.; Gong, C.; Zhang, L. Event-Triggered Nonlinear Model-Predictive Control for Optimal Ascent Guidance. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 7771–7784. [CrossRef]
79. Yang, B.; Li, Z.; Li, J.; Mo, Y.; Yan, J. Continuous-Time-Constrained Model Predictive Control with a Parallel Solver. *IEEE Trans. Contr. Syst. Technol.* **2025**, *33*, 845–857. [CrossRef]
80. Frekhaug, T.A.; Sanjurjo-Rivo, M.; Soler, M. Output-Feedback Tube Model Predictive Control for Asteroid Circumnavigation and Descent. *J. Guid. Control Dyn.* **2023**, *46*, 2235–2244. [CrossRef]
81. Mammarella, M.; Capello, E.; Lorenzen, M.; Dabbene, F.; Allgower, F. A general Sampling-Based SMPC Approach to Spacecraft Proximity Operations. In *Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*; IEEE: Melbourne, Australia, 2017. [CrossRef]
82. Liu, C.; Yang, H.; Li, S.; Li, J. Convex Optimization of Stochastic Path-Constrained Trajectories Near Asteroids. *Aerosp. Sci. Technol.* **2024**, *153*, 109463. [CrossRef]
83. Mammarella, M.; Capello, E.; Park, H.; Guglieri, G.; Romano, M. Tube-Based Robust Model Predictive Control for Spacecraft Proximity Operations in the Presence of Persistent Disturbance. *Aerosp. Sci. Technol.* **2018**, *77*, 585–594. [CrossRef]
84. Sun, W.; Pan, Y.; Lim, J.; Theodorou, E.A.; Tsiotras, P. Min-Max Differential Dynamic Programming: Continuous and Discrete Time Formulations. *J. Guid. Control Dyn.* **2018**, *41*, 2568–2580. [CrossRef]
85. Ozaki, N.; Campagnola, S.; Funase, R.; Yam, C.H. Stochastic Differential Dynamic Programming with Unscented Transform for Low-Thrust Trajectory Design. *J. Guid. Control Dyn.* **2018**, *41*, 377–387. [CrossRef]
86. Benedikter, B.; Zavoli, A.; Wang, Z.; Pizzurro, S.; Cavallini, E. Convex Approach to Covariance Control with Application to Stochastic Low-Thrust Trajectory Optimization. *J. Guid. Control Dyn.* **2022**, *45*, 2061–2075. [CrossRef]

87. Zhao, Z.; Shang, H.; Yu, Z.; Ren, J. Stochastic Trajectory Planning for Autonomous Aerobraking Using Convex Optimization and Covariance Control. *J. Guid. Control Dyn.* **2024**, *47*, 1521–1541. [CrossRef]
88. Zhao, J.; Chen, M. Dynamic Event-Triggered Robust Feedback Model Predictive Tracking Control of Air-Breathing Hypersonic Vehicle Based on Disturbance Preview. *IEEE Trans. Aerosp. Electron. Syst.* **2025**, *61*, 3291–3305. [CrossRef]
89. Huang, X.; Li, S.; Yang, H. Robust Intelligent Guidance Method for Distant Retrograde Orbit Station-Keeping. *J. Guid. Control Dyn.* **2026**, *49*, 374–386. [CrossRef]
90. Zhou, Y.; Van Kampen, E.-J.; Chu, Q. Incremental Approximate Dynamic Programming for Nonlinear Adaptive Tracking Control with Partial Observability. *J. Guid. Control Dyn.* **2018**, *41*, 2554–2567. [CrossRef]
91. Chai, R.; Tsourdos, A.; Savvaris, A.; Chai, S.; Xia, Y.; Chen, C.L.P. Six-DOF Spacecraft Optimal Trajectory Planning and Real-time Attitude Control: A Deep Neural Network-Based Approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 5005–5013. [CrossRef] [PubMed]
92. Yang, H.; Yan, J.; Li, S. Fast Computation of the Jovian-Moon Three-Body Flyby Map Based on Artificial Neural Networks. *Acta Astronaut.* **2022**, *193*, 710–720. [CrossRef]
93. Yan, J.; Yang, H.; Li, S. ANN-Based Method for Fast Optimization of Jovian-Moon Gravity-Assisted Trajectories in CR3BP. *Adv. Space Res.* **2022**, *69*, 2865–2882. [CrossRef]
94. Zhao, Y.; Yang, H.; Li, S. Real-Time Trajectory Optimization for Collision-Free Asteroid Landing Based on Deep Neural Networks. *Adv. Space Res.* **2022**, *70*, 112–124. [CrossRef]
95. Hu, J.; Liu, Y.; Yang, H.; Li, S. Fast Generation of Collision-Free Low-Thrust Trajectories for Asteroid Landing Using Deep Neural Networks. *Aerosp. Sci. Technol.* **2024**, *151*, 109316. [CrossRef]
96. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2016**. [CrossRef]
97. Hu, J.; Yang, H.; Li, S.; Zhao, Y. Densely Rewarded Reinforcement Learning for Robust Low-Thrust Trajectory Optimization. *Adv. Space Res.* **2023**, *72*, 964–981. [CrossRef]
98. Hu, J.; Yang, H.; Li, S.; Baoyin, H. Robust Design of Low-Thrust Gravity-Assist Trajectories via Reinforcement Learning. *J. Guid. Control Dyn.* **2026**, *49*, 1773–1788. [CrossRef]
99. Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Knoll, A. A Review of Safe Reinforcement Learning: Methods, Theories, and Applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 11216–11235. [CrossRef] [PubMed]
100. Borkar, V.; Jain, R. Risk-Constrained Markov Decision Processes. *IEEE Trans. Autom. Control* **2014**, *59*, 2574–2579. [CrossRef]
101. Mersha, M.; Lam, K.; Wood, J.; AlShami, A.K.; Kalita, J. Explainable Artificial Intelligence: A Survey of Needs, Techniques, Applications, and Future Direction. *Neurocomputing* **2024**, *599*, 128111. [CrossRef]
102. Santoso, F.; Garratt, M.A.; Anavatti, S.G. State-of-the-Art Integrated Guidance and Control Systems in Unmanned Vehicles: A Review. *IEEE Syst. J.* **2021**, *15*, 3312–3323. [CrossRef]
103. Padhi, R.; Chawla, C.; Das, P.G. Partial Integrated Guidance and Control of Interceptors for High-Speed Ballistic Targets. *J. Guid. Control Dyn.* **2014**, *37*, 149–163. [CrossRef]
104. Padhi, R.; Rakesh, P.R.; Venkataraman, R. Formation Flying with Nonlinear Partial Integrated Guidance and Control. *IEEE Trans. Aerosp. Electron. Syst.* **2014**, *50*, 2847–2859. [CrossRef]
105. Song, H.; Zhang, T. Fast Robust Integrated Guidance and Control Design of Interceptors. *IEEE Trans. Contr. Syst. Technol.* **2016**, *24*, 349–356. [CrossRef]
106. Qiao, Y.; Li, S.; Xin, M. Fixed-Time Air-Ground Cooperative Time-Varying Formation-Containment Control of Heterogeneous Multi-Agent Systems with Well-Informed Followers. *Robot. Auton. Syst.* **2026**, *198*, 105225. [CrossRef]
107. Zhang, Y.; Zhang, R.; Li, H. Online Path Decision of No-Fly Zones Avoidance for Hypersonic Vehicles Based on a Graph Attention Network. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *59*, 5554–5567. [CrossRef]
108. Zeng, X.; Yuan, R.; Yang, L.; Huang, X.; Li, S. Long-Term Multi-Region Observation Scheduling for Large-Scale Constellations via Two-Stage Hybrid Planning. *Aerosp. Sci. Technol.* **2026**, *169*, 111358. [CrossRef]


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.