applied sciences

MDPI

Article

Cutting-Edge Trajectory Optimization through Quantum Annealing

Andrea Carbone *, Federico De Grossi and Dario Spiller

Schoolof Aerospace Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, Italy; federico.degrossi@uniroma1.it (F.D.G.); dario.spiller@uniroma1.it (D.S.)
* Correspondence: and.carbone@uniroma1.it

Abstract: This paper introduces an innovative approach to explore the capabilities of Quantum Annealing (QA) for trajectory optimization in dynamic systems. The proposed method involves transforming trajectory optimization problems into equivalent binary optimization problems using Quadratic Unconstrained Binary Optimization (QUBO) representation. The procedure is general and adaptable, making it applicable to a wide range of optimal control problems that entail the satisfaction of dynamic, boundary, and path constraints. Specifically, the trajectory is discretized and approximated using polynomials. In contrast to the conventional approach of determining the polynomial degree (n) solely based on the number of boundary conditions, a specific factor is introduced in our method to augment the polynomial degree. As a result, the ultimate polynomial degree is calculated as a composite of two components: $n = l + (m - 1)$, where m denotes the count of boundary conditions and l signifies the number of independent variables. By leveraging inverse dynamics, the control required to follow the approximated trajectory can be determined as a linear function of independent variables l. As a result, the optimization function, which is represented by the integral of the square of the control, can be formulated as a QUBO problem and the QA is employed to find the optimal binary solutions.

Keywords: trajectory optimization; polynomial approximation; inverse dynamics; QUBO formulation; quantum annealing

Citation: Carbone, A.; De Grossi, F.; Spiller, D. Cutting-Edge Trajectory Optimization through Quantum Annealing. *Appl. Sci.* **2023**, *13*, 12853. https://doi.org/10.3390/app132312853

Academic Editors: Rosario Pecora and Lei Wang

Received: 20 September 2023
Revised: 20 November 2023
Accepted: 27 November 2023
Published: 30 November 2023

1. Introduction

Trajectory optimization plays a pivotal role across various domains, including space exploration and satellite missions, where finding the most efficient trajectories is essential for mission success, fuel conservation, and resource optimization. Recent advancements in optimization techniques have been propelled by emerging artificial intelligence technologies, notably Genetic Algorithms (GA) and Neural Networks (NN).

In the field of satellite constellations, Savitri et al. employed GA in [1] to optimize trajectories, maximizing coverage while minimizing revisit time. Rughani et al. in [2] utilized GA to optimize orbital trajectories for spacecraft swarms, facilitating collaborative tasks in space and collision avoidance. Particle Swarm Optimization demonstrated its prowess in [3], addressing real-time guidance for autonomous lunar landings and hazard avoidance, with promising experimental results. In [4], Particle Swarm Optimization was exploited to maximize asteroid surface coverage using hopping trajectories in low-gravity environments. Application to asteroids Itokawa and Bennu showed significant surface coverage improvement. A comprehensive overview of genetic algorithms was presented in [5], providing insights into established algorithms, their implementations, and associated pros and cons. Neural Network techniques have also made significant contributions. Federici et al. applied reinforcement learning in [6] to autonomously guide a spacecraft during a mission to impact a binary asteroid, utilizing neural networks for optical observation processing and achieving a high success rate despite uncertainties. Scorsoglio et al. in [7] leveraged reinforcement meta-learning, combining hazard detection and avoidance with image and radar data to ensure safe landing points. Addressing

challenges in asteroid exploration, Jiang et al. in [8] proposed the use of hopping rovers and explored deep reinforcement learning for 3D terrain path planning. Gaudet et al. in [9] focused on optimizing an asteroid hovering controller through reinforcement meta-learning, enabling stable positioning even with less precise asteroid data. Combination of these methods can yield advantageous outcomes, as seen in [10,11], where frameworks synergized particle swarm optimization for trajectory optimization and extreme learning machines for precise gravitational acceleration approximation from asteroids.

The advancements in this field have unlocked the potential for more advanced and efficient trajectory optimization methods, emphasizing the role of computational efforts in realizing these breakthroughs. As a result, the importance of computer processing power has become paramount in the field of trajectory optimization.Especially the computational limitations have become a significant challenge in handling complex trajectory design problems. Traditional computational resources struggle to handle the massive calculations required for optimal trajectory design, especially for scenarios involving intricate navigation through varying gravitational fields and orbital constraints.

Quantum Computing (QC) has demonstrated the ability to outperform classical and more commonly used optimizers in specific cases [12]. Some problems where QC provides an advantage over classical algorithms are factorization of prime numbers [13], unstructured database search [14],linear systems of equations [15], and simulation of quantum systems [16–18]. Research is active, with many algorithms proposed in many different areas, such as differential equations [19], artificial intelligence and machine learning [20,21] and optimization [22,23].

While the aforementioned literature focuses primarily on gate-based quantum computing, which operates by applying a series of gates to manipulate qubits, similar to classical computers, another notable quantum computing model is QA. QA exploits and guides the natural evolution of a quantum system to solve combinatorial optimization problems. During QA, the system gradually transitions from an initial state to a low-energy state, similar to the cooling process in metallurgy, hence the term "annealing". QA promises more efficient optimization solutions, especially for large-scale and complex problems [24,25]. In particular, current quantum annealers can handle computations with about an order of magnitude higher number of qubits than gate-based quantum machines. Proposed applications of QA include factorization of prime numbers [26], systems of polynomial equations [27], and differential equations [28,29].

The research is also exploring the application of quality control to practical engineering challenges, despite the limitations of current quantum annealers. In aerospace engineering, some applications have been proposed, such as addressing some artificial intelligence problems framed as combinatorial tasks [30], optimizing aircraft trajectories to prevent conflicts [31] and allowing agile planning of the Earth observation maximization of image acquisition [32].

The goal of this study is to explore the cutting-edge capabilities of QA and its potential applications. Although QA has garnered considerable attention in the optimization community, its potential in the specific context of space trajectory optimization has yet to be fully explored. The paper presents a transcription procedure that converts trajectory optimization problems into equivalent binary optimization problems using the QUBO representation. This versatile procedure can be applied to a wide range of optimal control problems, encompassing dynamic, boundary, and path constraints. Specifically, the trajectory is discretized and approximated using polynomials for each component. By employing inverse dynamics, the required control to follow the approximated trajectory can be determined. Consequently, the optimization function becomes the system's energy, represented by the integral of the square of the control. As this function depends on the form of the polynomial, the focus lies on optimizing the polynomial itself. This novel approach demonstrates significant potential for efficiently exploring extensive solution spaces, making it particularly promising for addressing complex trajectory design challenges in fields like space exploration and satellite missions.

This paper is organized as follows. With the purpose of providing the appropriate context and establishing the necessary nomenclature, Section 2 provides a brief description of the basic principle behind Quantum Annealig. Section 3 presents an overview of the transcription procedure for trajectory optimization, elucidating the mathematical formulation of the problem, including the cost function and various types of constraints. Moving on to Section 4, the lunar landing and rendezvous problems and their simplifications are reported. The results of the transcription procedure for this specific problem are discussed in Section 5, with an emphasis on specific considerations and limitations. Finally, in Section 6, conclusions and potential future research directions in the field of QA for trajectory optimization are presented.

2. Quantum Annealing Optimization

The fundamental principle behind QA is to encode the optimization problem into the energy states of a quantum system. For this purpose, let us briefly recall some fundamental concepts of quantum physics: a quantum system can have discrete levels of energy, among which the one with minimum energy is called the ground state. The energy of a quantum system is represented by its Hamiltonian operator H, which is closely related to the time evolution of the system through the famous Schrödinger Equation [33]. It is also possible to write a so-called eigenvalues problem of the Hamiltonian, as shown in Equation (1); this equation is sometimes called the Time Independent Schrödinger Equation. $|\psi_n\rangle$ (written here in the Dirac notation $|\cdot\rangle$, used to indicate quantum states) are called eigenvectors (or eigenstates) of H, and E_n are the eigenvalues that correspond to the energy values that the system can assume for $n = \{0, 1, 2, \dots\}$. Therefore, an optimization problem encoded into the Hamiltonian of a quantum system has the possible values of the cost function corresponding to the energy values E_n of the Hamiltonian, with E_0, the ground state, being the global minimum of the cost function.

$$H|\psi_n\rangle = E_n|\psi_n\rangle. \tag{1}$$

Quantum annealers are typically represented by a network of qubits, for which the Hamiltonian assumes the form of a matrix. Qubits are the quantum equivalent of classical bits, but unlike classical bits, which can only represent zero or one, qubits can exist in the superposition of both states simultaneously. This allows QA exploration of multiple possible solutions to the optimization problem simultaneously. The optimization process in QA involves the adiabatic theorem from quantum mechanics, which states that a system in the ground state, whose Hamiltonian is slowly time-varying, remains in the ground state of the instantaneous Hamiltonian. Therefore, for every instant of time t from zero to annealing time T, the following equation holds:

$$H(t)|\psi_0(t)\rangle = E_0(t)|\psi_0(t)\rangle. \tag{2}$$

The optimization procedure starts by preparing the system in the ground state of a simple Hamiltonian, called driver Hamiltonian (H_D); the optimization problem is instead encoded in the energy eigenvalues of a problem Hamiltonian, H_P; then, the system is evolved adiabatically from H_D to H_P [30] in such a way that the system state at the end of the evolution is the ground state of the problem Hamiltonian. At this point, to obtain the solution, we need to only operate a measurement of the qubtis [34]. Equation (3) shows a generic transition between the Dirver and Problem Hamiltonian, where $A(t/T)$ and $B(t/T)$ are arbitrary functions such that $A(0) = 1$, $B(0) = 0$ and $A(1) = 0$, $B(1) = 1$. In Equation (4), the state of the system is shown by the eigenvalues equation at the start and at the end of the annealing process, where $\psi_{0,D}$ stands for the ground state of the Driver Hamiltonian and $\psi_{0,P}$ is the ground state of the Problem Hamiltonian, the solution of the optimization problem.

$$H(t) = A(t/T)H_D + B(t/T)H_P, \tag{3}$$

$$H_D|\psi_{0,D}\rangle = E_{0,D}|\psi_{0,D}\rangle \xrightarrow{t=[0,T]} H_P|\psi_{0,P}\rangle = E_{0,P}|\psi_{0,P}\rangle. \quad (4)$$

To be solved by QA, an optimization problem must have a precise structure given by the Ising or QUBO formulations [35], which are described below.

- ***Ising Formulation*** revolves around determining the minimum energy configuration of a system of interacting spins. Let us consider an Ising model with n spins denoted as $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, where each spin can be in one of two states: +1 (up) or −1 (down). The energy of the Ising model is expressed by the following equation:

$$J_I(\sigma) = \sum_i a_i \sigma_i + \sum_{i<j} b_{ij} \sigma_i \sigma_j. \quad (5)$$

  In Equation (5), a_i represents the linear coupling coefficients for individual spins, while b_{ij} represents the coupling coefficients for pairs of spins. The first sum accounts for the energy associated with the individual spin states, whereas the second sum reflects the energy due to interactions between spins.
- ***QUBO Formulation*** is a powerful representation of optimization problems using a quadratic function of binary variables. We consider a problem with n binary variables denoted as $x = (x_1, x_2, \ldots, x_n)$, where each x_i can take either zero or one. The objective is to minimize the following quadratic function:

$$J_q(x) = x^T Q x. \quad (6)$$

  In Equation (6), x^T represents the transpose of the binary variable vector x, and Q is an $n \times n$ matrix of coefficients. The goal is to find the assignment of binary values to x that minimizes quadratic function $J_q(x)$. This formulation is widely used in QA to transform complex optimization problems into a format compatible with quantum systems.

The Ising and QUBO formulations are closely related through a simple transformation that maps Ising spins (σ_i) to binary variables (x_i). In the Ising Formulation, the linear coupling coefficients (a_i) are transformed into the diagonal components of matrix Q in the QUBO Formulation, while the coupling coefficients for pairs of spins (b_{ij}) are represented by the off-diagonal components. It is important to recognize that these formulations are mathematical abstractions designed to conveniently express optimization problems in a general format. When implementing an optimization problem on a real quantum annealer, the possible mismatch between the logical lattice of the problem (how the variables mathematically interact with each other) and the physical connectivity pattern (how the qubits are actually connected with each other in the annealer) must be taken into account. In fact, quantum annealers are built with a specific graph topology that dictates which qubits are physically connected with other qubits. If the logical graph of the problem does not match the physical graph of the machine, a procedure called embedding must be employed to adapt the first to the second. The embedding works by creating chains of qubits in the physical graph that are forced to behave as a single variable, always assuming the same value; through these chains, it is possible to connect qubits not physically connected in the annealer topology and crafting an embedded optimization problem equivalent to the starting one.

As an extremely simple schematic to illustrate the concept, in Figure 1,a logical graph is shown where three variables (represented as circles) interact as follows: Variable 1 interacts with Variables 3 and 4. On the other hand, the physical graph, on the right, does not provide a connection between Qubits 1 and 4. Therefore, a possible embedding is to create a chain including Qubits 2 and 4, such that Connection 1–4 can now be implemented exploiting Connections 1–2 and 2–4. We notice that, although the logical problem includes three variables, the embedded problem must use four qubits; therefore, the embedding procedure comes at the cost of increasing the number of qubits needed to tackle the problem.

As a consequence, a problem that requires embedding could require an annealer with a number of qubits much larger than the number of variables.

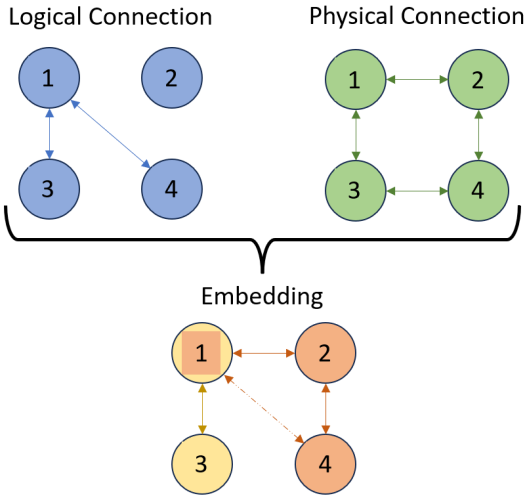

Figure 1. Scheme of a simple embedding procedure.

Moreover, utilizing longer chains in the embedding can increase susceptibility to noise and errors. Longer chains may become fragile, leading to potential disruptions that adversely affect the quality and reliability of the solution. Addressing these challenges is critical to unlocking the full potential of QA for solving complex optimization problems and realizing the transformative power of quantum computing.

One company that has made significant contributions in this area is D-Wave Systems, which is a leading company specializing in QA. Their quantum processors, known as quantum processing units (QPUs), are designed to perform QA computations with fully quantum systems (for more details, please refers to the following link: https://www.dwavesys.com/media/htjclcey/advantage_datasheet_v10.pdf, accessed on 6 November 2023). D-Wave's QPUs are used in various research and commercial applications to tackle real-world optimization challenges. D-Wave solvers are constantly being improved to address the limitations of quantum computing, and future advancements will likely include more sophisticated algorithms and improved processing capabilities to enable more efficient optimization.

3. Transcription Procedure

The goal of trajectory optimization is to find the optimal control inputs ($u(t)$) that steer a system to the desired final state (x_f) while considering the trade-offs between achieving the objective and minimizing the associated cost. Thus, given an initial condition (x_0) and a control input trajectory defined over finite interval $t \in [t_0, t_f]$, the cost function (J) in the Bolza form can be computed as follows:

$$J = \chi(\mathbf{x}(t_0), \mathbf{x}(t_f)) + \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t)dt, \tag{7}$$

where $\chi(\mathbf{x}(t_0), \mathbf{x}(t_f))$ is the cost component depending on initial state $\mathbf{x}(t_0)$ and final state $\mathbf{x}(t_f)$ of the system, and $\int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t)dt$ represents the integral of the instantaneous cost function $\mathcal{L}$ over the entire trajectory, which depends on state $\mathbf{x}(t)$, control input $\mathbf{u}(t)$, and time t. Moreover, the trajectory optimization problem must align with the underlying dynamics of the system. In the proposed method, the dynamic systems considered are linear or linearized; thus, the trajectory optimization problem can be expressed in the following comprehensive form:

$$\begin{aligned} \min_{\mathbf{u}(t)} \quad & \chi(\mathbf{x}(t_0), \mathbf{x}(t_f)) + \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t), \quad \forall t \in [t_0, t_f] \\ & \mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f. \end{aligned} \tag{8}$$

Equation (8) provides a concise representation of the trajectory optimization problem, encapsulating the goal of reaching a desired final state while considering the cost incurred along the trajectory, subject to the system's dynamic behavior and initial condition. In particular, $\mathbf{x} = [\mathbf{s}(t), \dot{\mathbf{s}}(t)]^T$ represents the state vector, resulting in a total of six components. This vector is composed of both position ($\mathbf{s}(t) = [x, y, z]^T$) and velocity ($\dot{\mathbf{s}}(t) = [v_x, v_y, v_z]^T$) along the three trajectory components.

The proposed transcription procedure aims to map the common trajectory optimization problem (Equation (8)) into a binary optimization problem in a QUBO form (Equation (6)), which can be effectively addressed using QA techniques. Central to the transcription procedure is the discretization of the trajectory by representing the state variable and its derivatives through polynomial functions across finite interval $t \in [t_0, t_f]$. In contrast to the usual practice of determining the polynomial degree (n) based on the number of boundary conditions, our method increases the polynomial degree by a specific factor. Consequently, the ultimate polynomial degree is obtained as the combination of two components: $n = l + (m - 1)$, where m represents the number of boundary conditions and l signifies the count of independent variables. These independent variables are essentially the coefficients of the polynomials that play a crucial role, acting as pivotal crossing points capable of altering the trajectory's shape. This core concept forms the foundation of the proposed optimization approach, where both the trajectory and its derivatives can be expressed as functions of l independent variables. In our procedure, we chose to minimize the number of variables, thus selecting $l = 1$, leading to polynomial degree n being simply equal to m. The decision to set $l = 1$ was primarily dictated by a desire to keep the approach simple and practical, making it easier to convert to the QUBO format. Moreover, it is worth noting that in other works, a similar choice was made to set "L" to 1 [3,36]. This approach is commonly adopted in trajectory optimization research for its practical advantages. Thus, the main goal of the article is not to add complexity to the underlying mathematics, but rather to test QA with entirely new challenges, such as trajectory optimization for space applications. Consequently, this work aims to take the initial step in exploring QA with an entirely novel method, and the first results are definitely promising.

By leveraging the polynomial approximations, the state vector and its derivatives can be explicitly written as a linear function of the free coefficient parameter:

$$\begin{aligned} \mathbf{s}(t) &= \boldsymbol{\alpha}_m t^m + \sum_{i=0}^{m-1} \boldsymbol{\alpha}_i t^i, \\ \frac{d^k \mathbf{s}(t)}{dt^k} &= \prod_{j=0}^{k-1} (m - j) \cdot \boldsymbol{\alpha}_m t^{(m-k)} + \sum_{i=1}^{m-1} \left(\prod_{j=0}^{k-1} (i - j)\right) \cdot \boldsymbol{\alpha}_i t^{(i-k)}. \end{aligned} \tag{9}$$

Vector $\boldsymbol{\alpha}_i = [\alpha_{i_x}, \alpha_{i_y}, \alpha_{i_z}]^T$ corresponds to the i-th set of fixed polynomial coefficients, while $\boldsymbol{\alpha}_m = [\alpha_{m_x}, \alpha_{m_y}, \alpha_{m_z}]^T$ represents the vector containing three independent parameters that influence the trajectory's components. By imposing the boundary conditions in Equation (9), we can obtain a system of m equations, where $\boldsymbol{\alpha}_i$ are considered unknowns and $\boldsymbol{\alpha}_m$ is considered a free parameter. As a consequence, all $\boldsymbol{\alpha}_i$ can be expressed as a linear function of $\boldsymbol{\alpha}_m$ and boundary conditions. By substituting the obtained expression $\boldsymbol{\alpha}_i$ in Equation (9), it becomes possible to obtain a linear expression as the one in Equation (10), where $\boldsymbol{\alpha}_m$ is isolated, and $M(t)$ and $N(t)$ are the expressions of two generic matrices as a function of time.

$$\begin{aligned} \mathbf{s}(t, \boldsymbol{\alpha}_m) &= M(t)\boldsymbol{\alpha}_m + N(t), \\ \frac{d^k \mathbf{s}(t, \boldsymbol{\alpha}_m)}{dt^k} &= \frac{d^k M(t)}{dt^k}\boldsymbol{\alpha}_m + \frac{d^k N(t)}{dt^k}. \end{aligned} \tag{10}$$

By using the general expression linear dynamical in Equation (8), it becomes feasible to derive control input $\mathbf{u}(t)$:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{s}} \\ \ddot{\mathbf{s}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \Lambda & H \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ \dot{\mathbf{s}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u}. \tag{11}$$

Control vector $\mathbf{u}(t)$ appears only in the second term of Equation (11), where $\mathbf{I}$ represents the identity matrix, while Λ and H are constant matrices. These matrices are typically specific to the dynamics of the problem at hand, but in the current discussion, they are considered as generic expressions without specific values or definitions. The explicit expression of $\mathbf{u}(t)$ can be formulated as follows:

$$\mathbf{u}(t) = (\ddot{\mathbf{s}} - H\dot{\mathbf{s}} - \Lambda\mathbf{s}). \tag{12}$$

Then, by substituting Equation (10) in Equation (12), control input $\mathbf{u}(t)$ is expressed as a linear function of $\boldsymbol{\alpha}_m$. However, from this point on, variable $\boldsymbol{\alpha}_m$ is represented with symbol $\boldsymbol{\alpha}$ to simplify the notation.

$$\begin{aligned} &\mathbf{u}(t) = P\boldsymbol{\alpha} + R, \\ \text{where: } &\begin{cases} P = (\ddot{M}(t) - H\dot{M}(t) - \Lambda M(t)) \\ R = (\ddot{N}(t) - H\dot{N}(t) - \Lambda N(t)), \end{cases} \end{aligned} \tag{13}$$

where P and R are functions of variables M, N, L, and H, and their expressions depend on both the system's dynamics and the specific polynomial chosen to approximate the trajectory. As of now, they are represented in a general form, but in practical applications, these variables are given explicit expressions tailored to the specific problem at hand. Let us suppose now that our cost function aims to minimize control, and we discretize the trajectory by generating vector of d time points $t = (t_0, t_1, \ldots, t_i, \ldots, t_{d-1}, t_d)$. As a result, the cost function transforms into a summation of the square of the control for each time instant, and it can be expressed as follows:

$$\begin{aligned} J &= \sum_{i=0}^{d} \mathbf{u}(t_i)^T \mathbf{u}(t_i) = \sum_{i=0}^{d} (P_i\boldsymbol{\alpha} + R_i)^T (P_i\boldsymbol{\alpha} + R_i), \\ &= \boldsymbol{\alpha}^T (\sum_{i=0}^{d} P_i^T P_i)\boldsymbol{\alpha} + 2(\sum_{i=0}^{d} R_i^T P_i)\boldsymbol{\alpha} + (\sum_{i=0}^{d} R_i^T R_i). \end{aligned} \tag{14}$$

Value $(\sum_{i=0}^{d} R_i^T R_i)$ can be safely disregarded since it remains constant. Its presence in a cost function is, therefore, inconsequential, as adding or subtracting a constant value from a cost function has no impact on the result. Then, by substituting $P_{tot} = (\sum_{i=0}^{d} P_i^T P_i)$ and $V_{tot} = (\sum_{i=0}^{d} R_i^T P_i)$ in Equation (14), the cost function can be succinctly formulated as

$$J = \boldsymbol{\alpha}^T P_{tot}\boldsymbol{\alpha} + 2V_{tot}\boldsymbol{\alpha}. \tag{15}$$

The next step to achieve the QUBO Formulation is to express each real variable in $\boldsymbol{\alpha}$ by a number of n_b binary variables. In executing this, the fixed-point binary representation is sought such that the maximum and minimum real values it can represent correspond to the boundaries of $\boldsymbol{\alpha}$. Indeed, referring to Equation (16), the real variable $\boldsymbol{\alpha}$ is equal to $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{min}$ when every $z_k = 0$, and $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{max}$, every $z_k = 1$.

$$\boldsymbol{\alpha} = \boldsymbol{\alpha}_{min} + G\mathbf{z} \quad \text{with } \mathbf{z} = \{0, 1\}, \tag{16}$$

$$G = \frac{1}{2^{n_b}-1}\begin{bmatrix} (\Delta a_x \cdot \mathbf{g}) & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & (\Delta a_y \cdot \mathbf{g}) & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & (\Delta a_z \cdot \mathbf{g}) \end{bmatrix}, \tag{17}$$

where $\boldsymbol{\alpha}_{min}$ and $\boldsymbol{\alpha}_{max}$ are the upper and lower boundaries, respectively, matrix G has the size of $n_{real} \times n_b n_{real}$ ($n_{real} = 3$), $\mathbf{g}$ has the size of n_b and represents the binary vector used for the binary form representation of integers, defined as $\mathbf{g} = (2^0, 2^1, \ldots, 2^{n_b-1})$. Additionally, $\Delta a_i = a_{max_i} - a_{min_i}$ (where $i = x, y, z$) denotes the difference between the upper and lower limits of the coefficient for each respective component. By substituting Equation (16) into Equation (15), an expression of the cost function in terms of binary variables $\mathbf{z}$ is achieved (Equation (18)), where L is the matrix of quadratic terms, $\bar{m}$ is the vector of linear terms, and N is a constant. By exploiting the fact that, for binary variables, $\mathbf{z}^T\mathbf{z} = \mathbf{z}$, the linear term is included into the quadratic matrix, leading to the formulation on the right hand side of Equation (18), with $Q = \bar{Q} + \mathrm{diag}(L)$; furthermore, constant N can be ignored for the purpose of optimization.

$$\begin{gathered} J = \mathbf{z}^T\bar{Q}\mathbf{z} + L\mathbf{z} + N \quad \Rightarrow \quad J = \mathbf{z}^T Q \mathbf{z}, \\ \begin{cases} \bar{Q} = G^T P_i G \\ L = 2\left(\boldsymbol{\alpha}_{min}^T P_{tot} + V_{tot}\right) G \\ N = \boldsymbol{\alpha}_{min}^T P_{tot} \boldsymbol{\alpha}_{min} + 2 V_{tot} \boldsymbol{\alpha}_{min} \end{cases}. \end{gathered} \tag{18}$$

The above-described methodology allows to formulate a linearized trajectory optimization as a QUBO problem, with the objective of minimizing the control employed and including the dynamical and boundaries constraints. The proposed approach offers a versatile and comprehensive solution for addressing a wide spectrum of linear optimal control problems. By transcending specific case limitations, it introduces a unified strategy that is adaptable to diverse challenges.

4. Lunar Landing and Rendezvous Applications

The main objective of both of these mission applications is to navigate the spacecraft in such a way that it can be safely and accurately landed or docked at a specified predetermined location. In achieving this goal, it is crucial that the spacecraft reaches this final position with specific attributes simultaneously satisfied. These attributes include having both zero relative velocity and zero relative acceleration. This ensures a controlled and precise touchdown or a docking operation, minimizing the potential for any sudden or undesirable movements during the final phases of the mission. This objective can be cast in the form of a boundary condition problem. Notably, a total of five boundary conditions ($m = 5$) prove sufficient to comprehensively define the entirety of this problem:

$$\begin{gathered} \mathbf{s}(t_0) = \mathbf{s}_0 \quad \dot{\mathbf{s}}(t_0) = \dot{\mathbf{s}}_0 \\ \mathbf{s}(t_f) = \mathbf{s}_f \quad \dot{\mathbf{s}}(t_f) = 0 \quad \ddot{\mathbf{s}}(t_f) = 0 \end{gathered}. \tag{19}$$

Consequently, the trajectory can be approximated by a fifth-degree polynomial (as elaborated in Section 3).

$$\mathbf{s}(t) = \boldsymbol{\alpha}_0 + \boldsymbol{\alpha}_1 t + \boldsymbol{\alpha}_2 t^2 + \boldsymbol{\alpha}_3 t^3 + \boldsymbol{\alpha}_4 t^4 + \boldsymbol{\alpha}_5 t^5. \tag{20}$$

In (20), five coefficients of each component are explicitly determined, while one coefficient (for each component) remains unconstrained and serves as the adjustable parameter (in this instance, $\boldsymbol{\alpha}_2 = [\alpha_{2_x}, \alpha_{2_y}, \alpha_{2_z}]$), representing the variable to be optimized through QA. By including Equation (19) in Equation (20), the coefficient values can be expressed as a function of boundary conditions and the free parameters:

$$\begin{cases} \boldsymbol{\alpha}_0 = \mathbf{s}_0 \\ \boldsymbol{\alpha}_1 = \dot{\mathbf{s}}_0 \\ \boldsymbol{\alpha}_3 = (10\Delta\mathbf{s} - 6\dot{\mathbf{s}}_0 t_f - 3\boldsymbol{\alpha}_2 t_f^2) \ / \ t_f^3 \\ \boldsymbol{\alpha}_4 = (-15\Delta\mathbf{s} + 8\dot{\mathbf{s}}_0 t_f + 3\boldsymbol{\alpha}_2 t_f^2) \ / \ t_f^4 \\ \boldsymbol{\alpha}_5 = (6\Delta\mathbf{s} - 3\dot{\mathbf{s}}_0 t_f - \boldsymbol{\alpha}_2 t_f^2) \ / \ t_f^5 \end{cases}, \tag{21}$$

where $\Delta\mathbf{s} = \mathbf{s}_f - \mathbf{s}_0$ represents the difference between the terminal and initial vector positions of the trajectory. By including Equation (21) into Equation (20), the spacecraft's trajectory can be explicitly represented as a function of $\boldsymbol{\alpha}_2$.

$$\mathbf{s}(t, \boldsymbol{\alpha}_2) = M\boldsymbol{\alpha}_2 + \hat{N}^T T = M\boldsymbol{\alpha}_2 + N,$$

$$\text{where:} \quad \begin{cases} M = \begin{bmatrix} D^T T & 0 & 0 \\ 0 & D^T T & 0 \\ 0 & 0 & D^T T \end{bmatrix} \\ T = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 & t^5 \end{bmatrix}^T \\ D = \begin{bmatrix} 0 & 0 & 1 & -\frac{3}{t_f} & \frac{3}{t_f^2} & -\frac{1}{t_f^3} \end{bmatrix}^T \\ \hat{N} = \begin{bmatrix} \mathbf{s}_0 & \dot{\mathbf{s}}_0 & 0 & \frac{10\Delta\mathbf{s} - 6\dot{\mathbf{s}}_0 t_f}{t_f^3} & \frac{-15\Delta\mathbf{s} + 8\dot{\mathbf{s}}_0 t_f}{t_f^4} & \frac{6\Delta\mathbf{s} - 3\dot{\mathbf{s}}_0 t_f}{t_f^5} \end{bmatrix}^T \end{cases}. \tag{22}$$

4.1. Lunar Landing Problem

The mathematical expressions governing the dynamics of a lunar lander can be derived using Newton's law in the context of no central forces. In this context, the gravitational influence and the propulsive thrust produced by the propulsion system act as the only forces of motion [37]. Furthermore, a flat ground is considered, thus obtaining linear dynamics. This preference is logically valid, since the present study limits its analysis to only the terminal phase of the landing trajectory. Accordingly, the formulation of the dynamics can be written as follows:

$$\ddot{\mathbf{s}}(t) = \mathbf{g}_M + \mathbf{u}(t), \tag{23}$$

where the unvarying gravitational acceleration vector of the Moon, $\mathbf{g}_M = [0, 0, -g_M]^T$, exclusively acting along the z-axis ($g_M = 1.62509$ m/s^2), along with control thrust and acceleration vectors $\mathbf{T}$ and $\mathbf{u}$, respectively, contribute to the movement. Upon obtaining the second derivative of Equation (22), it becomes evident that solely time vector T undergoes differentiation, while the other terms remain held constant. The resulting $\ddot{\mathbf{s}}(t, \boldsymbol{\alpha}_2)$ can be included in Equation (23) to find the expression of control vector $\mathbf{u}$:

$$\mathbf{u}(t) = \ddot{\mathbf{s}}(t, \boldsymbol{\alpha}_2) - \mathbf{g}_M = \ddot{M}\boldsymbol{\alpha}_2 + \ddot{N} - \mathbf{g}_M = P\boldsymbol{\alpha}_2 + R, \tag{24}$$

where $R = \ddot{N} - \mathbf{g}_M$ represents the component that remains independent of the variable optimization. Then, by following the steps reported from Equation (15) to Equation (18) the QUBO problem can be derived.

4.2. Rendezvous Problem

The Clohessy–Wiltshire equations are a set of linearized equations commonly used in orbital mechanics to describe the relative motion between two objects in close proximity during a rendezvous or docking maneuver. These equations provide an approximation of the relative dynamics in a simplified form, making it easier to analyze and plan maneuvers [38]. The Clohessy—Wiltshire equations are particularly useful when the relative distances between the two objects are small compared to the orbital distances. They are derived by linearizing the equations of motion around a reference orbit. The equations are typically used for missions where the spacecraft is performing relatively short-duration

maneuvers, such as rendezvous, docking, or formation flying. The basic form of the Clohessy–Wiltshire equations is as follows:

$$\begin{aligned}\ddot{\mathbf{s}}(t) &= \begin{bmatrix} 0 & 2n & 0 \\ -2n & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\mathbf{s}}(t) + \begin{bmatrix} 3n^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -n^2 \end{bmatrix} \mathbf{s}(t) + \mathbf{u}(t) \\ &= H\dot{\mathbf{s}}(t) + \Lambda\mathbf{s}(t) + \mathbf{u}(t),\end{aligned} \tag{25}$$

where $n = \sqrt{\mu/R}$ is the mean motion of the reference orbit, R is the radius of the target body's circular orbit, and μ is the standard gravitational parameter. In this application, the motion is around the Earth, thus μ = 398,600 km^3/s^2. From Equation (22), the polynomial approximation and its derivatives can be included in the dynamical formulation (Equation (25)) to find the expression of control vector $\mathbf{u}$:

$$\begin{aligned}\mathbf{u}(t) &= \ddot{\mathbf{s}}(t, \boldsymbol{\alpha}_2) - H\dot{\mathbf{s}}(t, \boldsymbol{\alpha}_2) - \Lambda\mathbf{s}(t, \boldsymbol{\alpha}_2) \\ &= \ddot{M}\boldsymbol{\alpha}_2 + \ddot{N} - H(\dot{M}\boldsymbol{\alpha}_2 + \dot{N}) - \Lambda(M\boldsymbol{\alpha}_2 + N) \\ &= (\ddot{M} - H\dot{M} - \Lambda M)\boldsymbol{\alpha}_2 + (\ddot{N} - H\dot{N} - \Lambda N) \\ &= P\boldsymbol{\alpha}_2 + R.\end{aligned} \tag{26}$$

The control vector (in Equation (26)) is a linear function of variable $B\alpha_2$ and it has the same form as provided in Equation (10). Thus, by following the transposition procedure from Equation (14) to Equation (18), once again, the QUBO Formulation of the problem can be derived.

5. Results

In this section, the empirical results obtained using the D-Wave Advantage 4.1 QPU through the D-Wave cloud service "Leap" are presented. This quantum annealer adheres to a topology named Pegasus, featuring a pool of approximately 5000 qubits for computational tasks. Within this topology, each qubit establishes connections with fifteen others, resulting in a total of around 35,000 interconnections. The main objective is to assess the QA's performance in terms of computational time and optimized cost function. This evaluation is carried out through a comparative analysis with established optimization techniques. Specifically, the Quantum Annealer's performance is compared with the Interior-Point (IP) optimization method provided by the "fmincon" function in MATLAB R2022b. Additionally, a comparison is made with the genetic algorithm, Particle Swarm Optimization (PSO), which is implemented using the "particleswarm" function on MATLAB [39]. Both the IP and the PSO method are implemented on a workstation equipped with 32 GB of RAM, an Intel Core i7-12700H 2.70 GHz CPU from the 12th Generation, and an NVIDIA GeForce RTX 3080 Ti GPU with 16 GB of dedicated RAM (manufactured by Intel and NVIDIA Corporation, Santa Clara, CA, USA).

The parameters used to set the simulations are provided in Table 1.

Table 1. Parameters to set the QUBO problem.

	Final Time (t_f)	Step	Parameters ($3 \times n_b$)	Upper Limit (α_{max})	Lower Limit (α_{min})
LL	100 s	0.1 s	96 qubit	[10, 10, 10]	−[10, 10, 10]
RV	320 s	0.1 s	96 qubit	[10, 10, 10]	−[10, 10, 10]

We note that the variables within the problem being examined are considerably fewer in number than what the physical constraints demand. Indeed, by choosing to use 32 bits to convert the three real components into binary, the number of variables of the problem are kept very low: 3×32, which amounts to 96 variables, while the non-zero components of the matrix Q count 1488. Regarding the embedding procedure, it is carried out automatically by the built-in algorithms of D-Wave. The resulting embeddings are easily found and have chain lengths of eight components at maximum. As a result, the

question of embedding does not arise as a concern and is resolved indirectly through the application of a substantially smaller-sized problem. The decision to limit the qubit count to 32 is not arbitrary but based on a systematic analysis of the cost functions in the context of both Lunar Landing and Rendezvous applications. Indeed, the number of qubits directly impacts the precision of conversion from real to binary numbers, as indicated in Equation (17). The outcomes of this analysis are showcased in Figure 2, where the cost functions are plotted against the base-2 logarithm of the number of qubits ($log_2(n_b)$). This visualization simplifies the presentation of results and underscores that a plateau is reached for both applications when $log_2(n_b)$ equals five, corresponding to $n_b = 32$. As such, pursuing a greater number of variables proves to be superfluous.

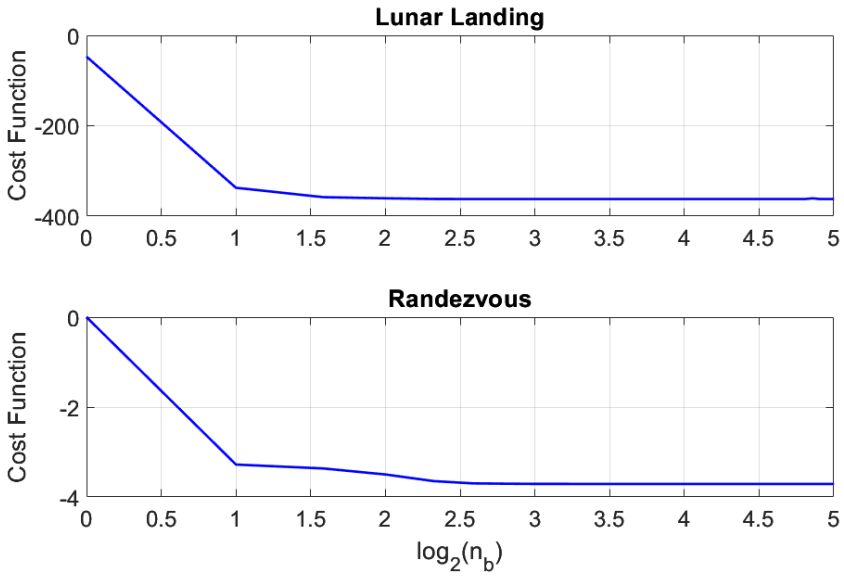

Figure 2. Lunar Landing and Rendezvous Cost Functions as a function of Number of Quibit.

In addition, a Monte Carlo simulation is carried out for each test case, utilizing 10,000 samples to provide a comprehensive evaluation of QA in comparison to IP and PSO, with a focus on assessing the achieved cost function and computational time. This extensive analysis allows for a thorough understanding of the relative performance of the methods in terms of optimization quality and efficiency.

5.1. Lunar Landing Results

In Table 2, the lunar landing parameters are provided. Specifically, the initial position is selected to uphold the validity of the approximation applied to the dynamics equations, which assumes the lunar surface to be flat. As a result, the spacecraft is situated at an altitude of 3 km. This initial configuration is based on a previously published works from 2022 [36,40], ensuring a consistent and valid reference point for our analysis.

Table 2. Parameters to set the Lunar Landing problem.

	Position (m)	**Velocity (m/s)**	**Acceleration (m^2/m)**
Initial Condition	[0, 0, 3000]	[0, 50, −10]	Unconstrained
Final Condition	[3000, 2000, 0]	[0, 0, 0]	[0, 0, 0]

The QPU results are shown in Figure 3, in terms of position, velocity, and acceleration. The figures readily demonstrate that boundary conditions are fully met; notably, the final position is reached with zero velocity and acceleration. This implies that the QA successfully yields an optimal solution for the problem at hand.

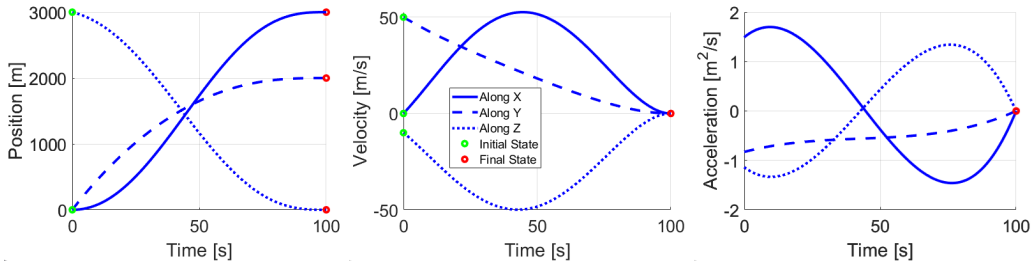

Figure 3. Solution of Lunar Landing by QPU.

The trajectory profiles and the control law are visually presented in Figures 4 and 5, featuring a comparative analysis with trajectories and control generated using MATLAB's PSO and IP methods, respectively. In both figures, the substantial overlap between trajectories offers compelling evidence of the quality of QA's solutions. Additionally, on the right side of both figures, the alignment of QA's control trajectory with those of the IP and PSO methods is illustrated, further affirming their striking similarity. This robustly demonstrates QA's effectiveness in tackling these specific problems.

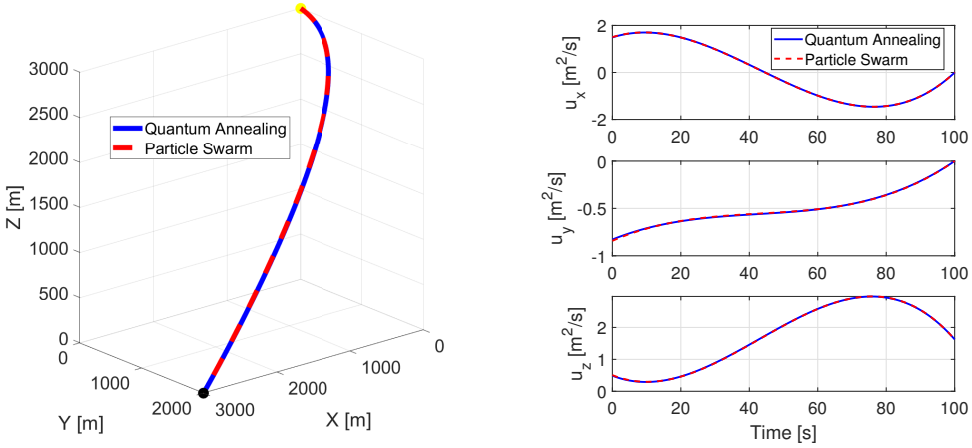

Figure 4. Comparison of trajectory and control law for Lunar Landing application between Quantum Annealing (blue line) and Particle Swarm Optimization (red line).

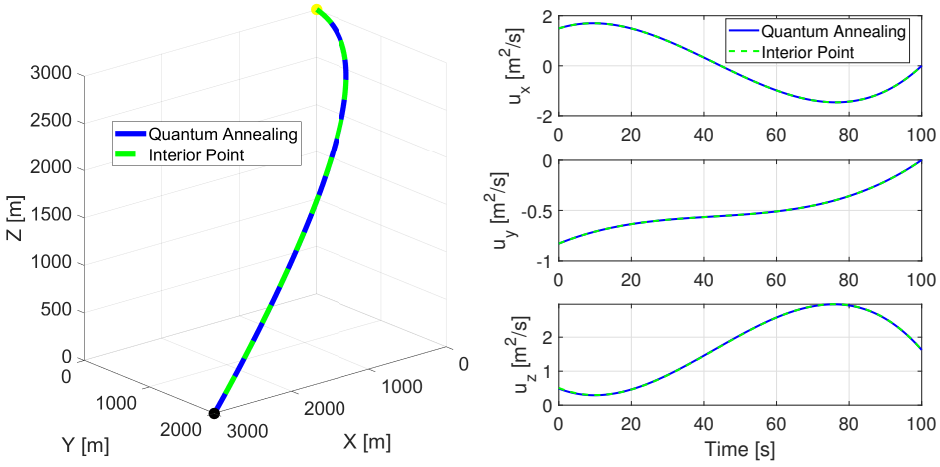

Figure 5. Comparison of trajectory and control law for Lunar Landing application between Quantum Annealing (blue line) and Interior Point (gree line).

Furthermore, a Monte Carlo simulation is executed employing 10,000 samples to comprehensively evaluate QA's performance in contrast to IP and PSO methods. The Monte Carlo comparison results are visually represented in Figure 6, encompassing computational

time and cost function optimization. In particular, the upper row reports the comparison between QA and PSO, while the bottom row reports the comparison between QA and IP methods. An insightful observation can be made when analyzing the results presented in Figure 6. Although the average value of the cost function is similar to those of QA (approximately −362.19), the PSO method (approximately −362.13), and the IP method (approximately −362.11), the QA method seems to exhibit a slightly better performance. Moreover, there is less data dispersion in QA's results. This indicates a more deterministic outcome compared to those of the other two methods. A significant difference is evident in terms of computational efficiency. Notably, the computational time required for QA (0.015 s) is an order of magnitude smaller than that of PSO (approximately 0.30 s) and IP methods (approximately 0.62 s).

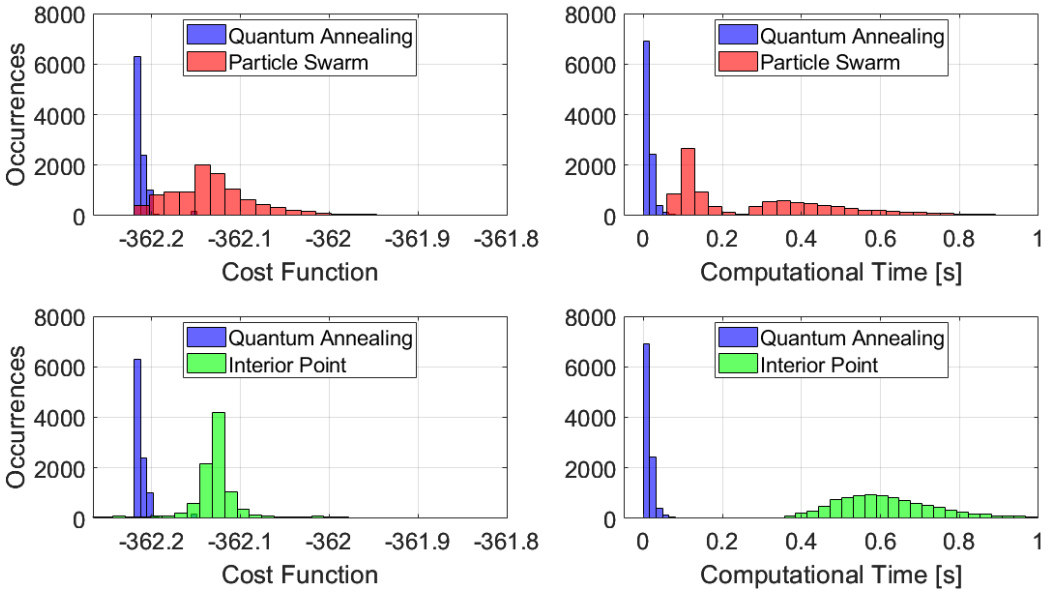

Figure 6. Comparison of Monte Carlo results for Lunar Landing application between QA (blue), PSO (red), and IP methods (green) in terms of computation time and cost function.

5.2. Rendezvous Results

Table 3 provides the parameters for the rendezvous scenario. Notably, the final conditions are uniformly set to zero, given the utilization of equations that pertain to relative motion. It is worth noting that this scenario was taken from the study detailed in [41], where the same polynomial approximation method is used to address a rendezvous problem. This ensures consistency and aligns with the methodology established in the scientific literature. In particular, the radius of the target spacecraft orbit is set to 773 km.

Table 3. Parameters to set the Rendezvous problem.

	Position (m)	**Velocity (m/s)**	**Acceleration (m^2/m)**
Initial Condition	[−10, −45, 3]	[0, 0, 0]	Unconstrained
Final Condition	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]

The outcomes in terms of position, velocity, and acceleration are showcased in Figure 7. Similar to the previous application, the rendezvous scenario successfully achieves the targeted boundary conditions, with all values equating to zero.

Figures 8 and 9 provide a comparative analysis with trajectories and control laws generated using MATLAB's IP and PSO methods, respectively. Once again, it is evident that the results obtained from QA are highly comparable to the outcomes achieved through IP and PSO methods.

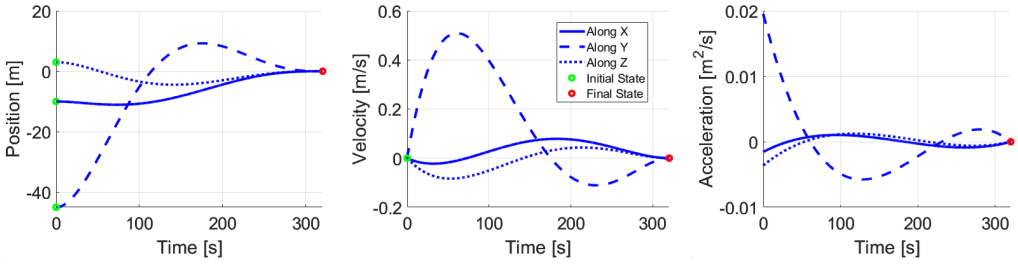

Figure 7. Solution of Rendezvous by QPU.

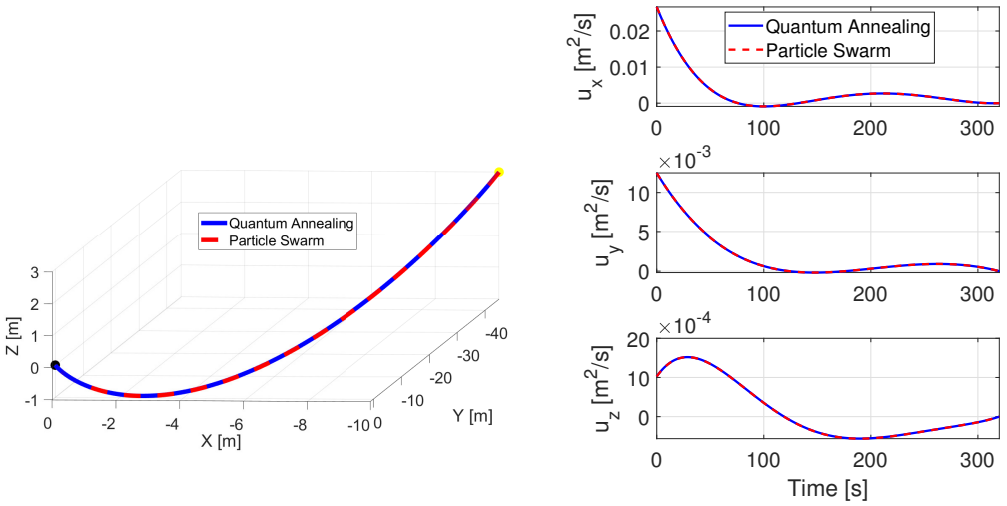

Figure 8. Comparison of trajectory and control law for Rendezvous application between Quantum Annealing (blue line) and Particle Swarm Optimization (red line).

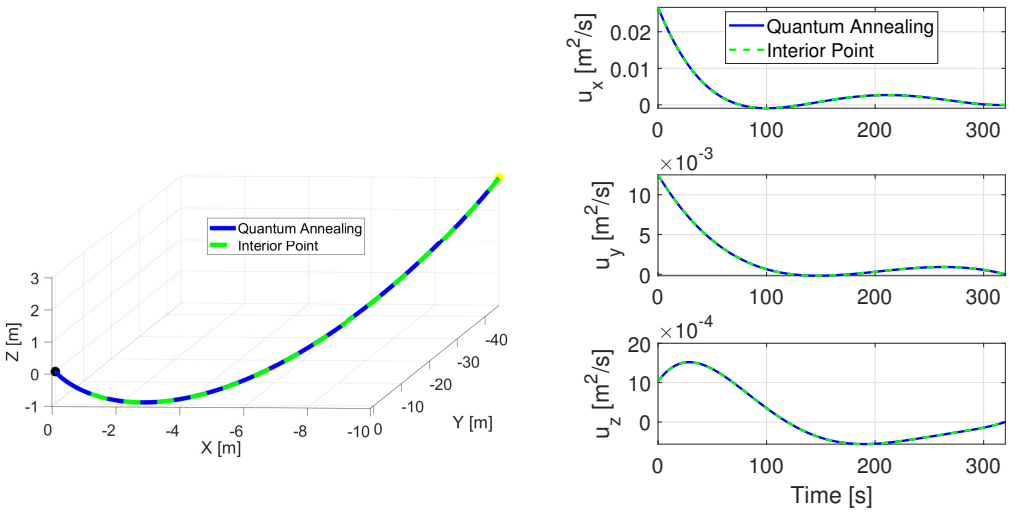

Figure 9. Comparison of trajectory and control law for Rendezvous application between Quantum Annealing (blue line) and Interior Point (gree line).

Additionally, a Monte Carlo simulation is conducted using 10,000 samples to thoroughly assess QA's performance in comparison to that of IP and PSO methods. The Monte Carlo comparison results, presented in Figure 10, encompass computational time and cost function optimization. Specifically, the upper row presents the comparison between QA and PSO, while the bottom row provides the comparison between QA and the IP method. Once again, the average values of the cost function are remarkably consistent across all three methods, with QA yielding approximately -0.370, IP yielding -0.361, and PSO yielding -0.361.

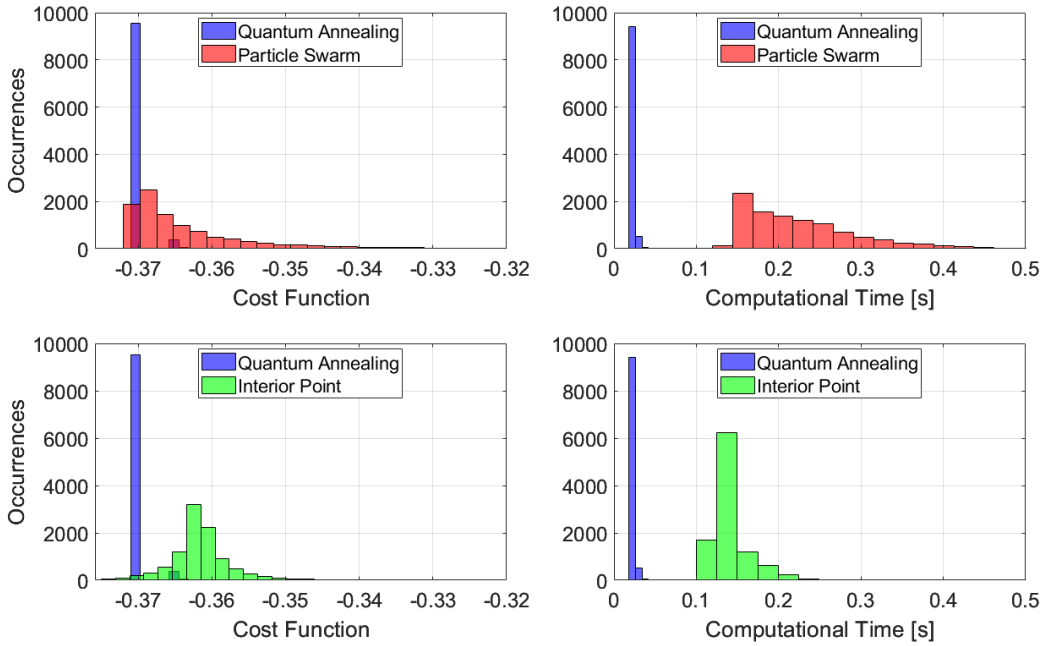

Figure 10. Comparison of Monte Carlo results for Rendezvous application between QA (blue), PSO (red), and IP methods (green) in terms of computation time and cost function.

Furthermore, there is notably less data dispersion in the results obtained using the QA method, indicating a higher degree of determinism compared to the other two approaches. Moreover, QA demonstrates an impressively low computational time of 0.023 s, even when applied in the context of the Rendezvous scenario. This computational efficiency is notably superior, being an order of magnitude lesser than that of the IP and PSO methods, which require approximately 0.141 and 0.226 s, respectively.

Upon analyzing the results presented across different scenarios, a consistent trend emerges. While the average cost function values for QA and fmincon demonstrate comparability, a remarkable supremacy of QA in terms of computational efficiency emerges.

6. Conclusions

This methodology leverages QA through a QUBO Formulation, translating trajectory optimization problems into the binary optimization domain. The proposed procedure presents a general approach that can be applied to optimization problems with linear dynamics. Its strength lies in its versatility, making it suitable for a wide range of scenarios within the realm of linear applications. Although it is designed for linear systems, its adaptability to different problems is a noteworthy advantage. Future research endeavors might focus on developing methods to tackle optimization challenges posed by nonlinear systems. Moreover, while the applications explored in this study may appear straightforward, a notable advantage in computational time is already discernible with the QA. This early advantage lays the groundwork for a deeper exploration of the potential harbored by this novel technology. The significance of the computational time disparity is indeed striking. The QA demonstrates a remarkable level of computational efficiency, evident through its computational time, which is orders of magnitude smaller than that of IP and PSO methods. This discrepancy in computation time may be a very important finding that could highlight an innate advantage of quantum computing in its ability to efficiently address optimization challenges. By exploiting the principles of quantum mechanics, QA is able to explore solution spaces in a parallel and probabilistic manner, enabling it to navigate through a vast number of potential solutions more quickly. In contrast, classical optimization methods often rely on iterative and sequential procedures that require greater computational efforts.

Nevertheless, it is worth noting that due to the polynomial-based approximation of the system state, an optimally exact solution might not be achieved. This approach strikes a

balance between optimizing for fuel consumption and computational efficiency. Opting for higher-degree polynomials could offer more flexibility for optimization, potentially yielding more accurate outcomes. However, this decision introduces a trade-off by potentially prolonging computational time, a factor undesirable in applications requiring swift computations. Hence, the selected approach reflects a compromise between result optimality and computational efficiency. Another limitation is that the flight time, Δt_f, has to be fixed. This is, again, due to the need for reducing the optimization into a QUBO form. Indeed, if t_f were a variable, it would introduce a multiplication factor with α in Equation (10). Consequently, non-linear terms would emerge, potentially reaching higher than second-degree terms within the cost function. This intricacy would render the transcription into a QUBO problem infeasible.

Given these encouraging outcomes and the numerous achievements in recent times, quantum computation has undeniably demonstrated its potential. Future research could enhance the methodology by exploring advanced trajectory approximation techniques to achieve greater precision while maintaining computational efficiency, such as B-Spline or Neural Network. In addition, the development of methods to handle nonlinear systems and to include time as a problem variable could expand the applicability of the methodology. Lastly, it might also be of interest to investigate the use of hybrid optimizers that combine quantum and classical optimization techniques, which could provide better solutions.

Author Contributions: Conceptualization, A.C. and F.D.G.; methodology, A.C. and F.D.G.; software, A.C. and F.D.G.; validation, A.C.; formal analysis, A.C.; investigation, A.C.; resources, D.S.; data curation, A.C.; writing—original draft preparation, A.C. and F.D.G.; writing—review and editing, A.C., F.D.G. and D.S.; visualization, A.C.; supervision, D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The subject of this paper was motivated by the research activity carried out in collaboration with CINECA (Daniele Ottaviani and Riccardo Mengoni).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Savitri, T.; Kim, Y.; Jo, S.; Bang, H. Satellite Constellation Orbit Design Optimization with Combined Genetic Algorithm and Semianalytical Approach. *Int. J. Aerosp. Eng.* **2017**, *2017*, 1235692. [CrossRef]
2. Rughani, R.; Barnhart, D. Using genetic algorithms for safe swarm trajectory optimization. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; p. 1916.
3. D'Ambrosio, A.; Carbone, A.; Spiller, D.; Curti, F. PSO-Based Soft Lunar Landing with Hazard Avoidance: Analysis and Experimentation. *Aerospace* **2021**, *8*, 195. [CrossRef]
4. Carbone, A.; Agostinelli, I.; D'Ambrosio, A.; Curti, F. Optimization of Hopping Trajectories for Asteroids Surface Exploration. In Proceedings of the 72nd International Astronautical Congress (IAC), Dubai, United Arab Emirates, 25–29 October 2021.
5. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
6. Federici, L.; Scorsoglio, A.; Ghilardi, L.; D'Ambrosio, A.; Benedikter, B.; Zavoli, A.; Furfaro, R. Image-Based Meta-Reinforcement Learning for Autonomous Guidance of an Asteroid Impactor. *J. Guid. Control Dyn.* **2022**, *45*, 2013–2028. [CrossRef]
7. Scorsoglio, A.; D'Ambrosio, A.; Ghilardi, L.; Furfaro, R.; Gaudet, B.; Linares, R.; Curti, F. Safe Lunar landing via images: A Reinforcement Meta-Learning application to autonomous hazard avoidance and landing. In Proceedings of the 2020 AAS/AIAA Astrodynamics Specialist Conference, Virtual, 9–13 August 2020; pp. 9–12.
8. Jiang, J.; Zeng, X.; Guzzetti, D.; You, Y. Path planning for asteroid hopping rovers with pre-trained deep reinforcement learning architectures. *Acta Astronaut.* **2020**, *171*, 265–279. [CrossRef]
9. Gaudet, B.; Linares, R.; Furfaro, R. Six degree-of-freedom hovering over an asteroid with unknown environmental dynamics via reinforcement learning. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; p. 0953.

10. D'Ambrosio, A.; Carbone, A.; Curti, F. Optimal Maneuvers Around Binary Asteroids Using Particle Swarm Optimization and Machine Learning. *J. Spacecr. Rocket.* **2023**, *60*, 1–15. [CrossRef]
11. D'Ambrosio, A.; Carbone, A.; Mastrofini, M.; Curti, F. Optimal Reference Orbit Tracking around Asteroids via Particle Swarm Optimization and Inverse Dynamics Technique. In Proceedings of the 31st AAS/AIAA Space Flight Mechanics Meeting (Charlotte 1st AAS/AIAA Space Flight Mechanics Meeting), Charlotte, NC, USA, 1–4 February 2021.
12. Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J.C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F.G.; Buell, D.A.; et al. Quantum supremacy using a programmable superconducting processor. *Nature* **2019**, *574*, 505–510. [CrossRef] [PubMed]
13. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
14. Grover, L.K. A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 212–219. [CrossRef]
15. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef]
16. Feynman, R.P. *Simulating Physics with Computers*; CRC Press: Boca Raton, FL, USA, 2018; pp. 133–153. [CrossRef]
17. Lloyd, S. Universal quantum simulators. *Science* **1996**, *273*, 1073–1078. [CrossRef]
18. Daley, A.J.; Bloch, I.; Kokail, C.; Flannigan, S.; Pearson, N.; Troyer, M.; Zoller, P. Practical quantum advantage in quantum simulation. *Nature* **2022**, *607*, 667–676. [CrossRef] [PubMed]
19. Lloyd, S.; De Palma, G.; Gokler, C.; Kiani, B.; Liu, Z.W.; Marvian, M.; Tennie, F.; Palmer, T. Quantum algorithm for nonlinear differential equations. *arXiv* **2020**, arXiv:2011.06571.
20. Dunjko, V.; Briegel, H.J. Machine learning and artificial intelligence in the quantum domain: A review of recent progress. *Rep. Prog. Phys.* **2018**, *81*, 074001. [CrossRef] [PubMed]
21. Cerezo, M.; Verdon, G.; Huang, H.Y.; Cincio, L.; Coles, P.J. Challenges and opportunities in quantum machine learning. *Nat. Comput. Sci.* **2022**, *2*, 567–576. [CrossRef]
22. Farhi, E.; Goldstone, J.; Gutmann, S. A quantum approximate optimization algorithm. *arXiv* **2014**, arXiv:1411.4028.
23. Hadfield, S.; Wang, Z.; O'gorman, B.; Rieffel, E.G.; Venturelli, D.; Biswas, R. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms* **2019**, *12*, 34. [CrossRef]
24. Kadowaki, T.; Nishimori, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E—Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.* **1998**, *58*, 5355–5363. [CrossRef]
25. Yulianti, L.P.; Surendro, K. Implementation of Quantum Annealing: A Systematic Review. *IEEE Access* **2022**, *10*, 73156–73177. [CrossRef]
26. Peng, X.; Liao, Z.; Xu, N.; Qin, G.; Zhou, X.; Suter, D.; Du, J. Quantum adiabatic algorithm for factorization and its experimental implementation. *Phys. Rev. Lett.* **2008**, *101*, 220405. [CrossRef]
27. Chang, T.H.; Lux, T.C.H.; Tipirneni, S.S. Least-squares solutions to polynomial systems of equations with quantum annealing. *Quantum Inf. Process.* **2019**, *18*, 374. [CrossRef]
28. Srivastava, S.; Sundararaghavan, V. Box algorithm for the solution of differential equations on a quantum annealer. *Phys. Rev. A* **2019**, *99*, 052355. [CrossRef]
29. Zanger, B.; Mendl, C.B.; Schulz, M.; Schreiber, M. Quantum algorithms for solving ordinary differential equations via classical integration methods. *Quantum* **2021**, *5*, 502. [CrossRef]
30. Smelyanskiy, V.N.; Rieffel, E.G.; Knysh, S.I.; Williams, C.P.; Johnson, M.W.; Thom, M.C.; Macready, W.G.; Pudenz, K.L. A near-term quantum computing approach for hard computational problems in space exploration. *arXiv* **2012**, arXiv:1204.2821.
31. Stollenwerk, T.; Ogorman, B.; Venturelli, D.; Mandra, S.; Rodionova, O.; Ng, H.; Sridhar, B.; Rieffel, E.G.; Biswas, R. Quantum Annealing Applied to De-Conflicting Optimal Trajectories for Air Traffic Management. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 285–297. [CrossRef]
32. Stollenwerk, T.; Michaud, V.; Lobe, E.; Picard, M.; Basermann, A.; Botter, T. Agile Earth Observation Satellite Scheduling with a Quantum Annealer. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 3520–3528. [CrossRef]
33. Krane, K.S. *Modern Physics*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
34. Farhi, E.; Goldstone, J.; Gutmann, S.; Sipser, M. Quantum computation by adiabatic evolution. *arXiv* **2000**, arXiv:quant-ph/0001106.
35. Glover, F.; Kochenberger, G.; Hennig, R.; Du, Y. Quantum bridge analytics I: A tutorial on formulating and using QUBO models. *Ann. Oper. Res.* **2022**, *314*, 141–183. [CrossRef]
36. Carbone, A.; Spiller, D.; Farissi, M.S.; Sasidharan, S.T.; Latorre, F.; Curti, F. Hardware-in-the-Loop Simulations of Future Autonomous Space Systems Aided by Artificial Intelligence. In Proceedings of the International Conference on Applied Intelligence and Informatics, Reggio Calabria, Italy, 1–3 September 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 83–99.
37. Wibben, D.R.; Furfaro, R.; Sanfelice, R.G. Optimal lunar landing and retargeting using a hybrid control strategy. In Proceedings of the 23rd AAS/AIAA Space Flight Mechanics Meeting, Spaceflight Mechanics 2013, Kauai, HI, USA, 10–14 February 2013; Univelt Inc.: Escondido, CA, USA, 2013; pp. 1901–1919.
38. Clohessy, W.; Wiltshire, R. Terminal guidance system for satellite rendezvous. *J. Aerosp. Sci.* **1960**, *27*, 653–658. [CrossRef]
39. The MathWorks Inc. *MATLAB Version 9.13.0 (R2022b)*; The MathWorks Inc.: Natick, MA, USA, 2010.

40. Spiller, D.; Carbone, A.; Latorre, F.; Curti, F. Hardware-in-the-loop Simulations of Remote Sensing Disaster Monitoring Systems with Real-Time On-Board Computation. In Proceedings of the 2022 IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE), Rome, Italy, 26–28 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 731–736.
41. Ventura, J.; Ciarcia, M.; Romano, M.; Walter, U. An inverse dynamics-based trajectory planner for autonomous docking to a tumbling target. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Diego, CA, USA, 4–8 January 2016; p. 0876. [CrossRef]