




## Article

# Comparative Study of Optimal Multivariable LQR and MPC Controllers for Unmanned Combat Air Systems in Trajectory Tracking

Alvaro Ortiz <sup>1</sup>, Sergio Garcia-Nieto <sup>2,\*</sup> and Raul Simarro <sup>2</sup>

<sup>1</sup> Escuela Técnica Superior de Ingeniería del Diseño, Universitat Politècnica de València, 46022 València, Spain; alormo1@etsid.upv.es

<sup>2</sup> Instituto Universitario de Automática e Informática Industrial, Universitat Politècnica de València, 46022 València, Spain; rausifer@isa.upv.es

\* Correspondence: sgnieto@isa.upv.es; Tel.: +34-963-877-007

**Abstract:** Guidance, navigation, and control system design is, undoubtedly, one of the most relevant issues in any type of unmanned aerial vehicle, especially in the case of military missions. This task needs to be performed in the most efficient way possible, which involves trying to satisfy a set of requirements that are sometimes in opposition. The purpose of this article was to compare two different control strategies in conjunction with a path-planning and guidance system with the objective of completing military missions in the most satisfactory way. For this purpose, a novel dynamic trajectory-planning algorithm is employed, which can obtain an appropriate trajectory by analyzing the environment as a discrete 3D adaptive mesh and performs a softening process a posteriori. Moreover, two multivariable control techniques are proposed, i.e., the linear quadratic regulator and the model predictive control, which were designed to offer optimal responses in terms of stability and robustness.

**Keywords:** unmanned aerial vehicles (UAV); algorithm; UAV control; tracking; Octree; mapping; sensors and actuators in UAVs; path planning; rectangloid; trajectory



**Citation:** Ortiz, A.; Garcia-Nieto, S.; Simarro, R. Comparative Study of Optimal Multivariable LQR and MPC Controllers for Unmanned Combat Air Systems in Trajectory Tracking. *Electronics* **2021**, *10*, 331. <https://doi.org/10.3390/electronics10030331>

Academic Editor: Rafael Casado, Aurelio Bermúdez and Mahmut Reyhanoglu

Received: 14 November 2020

Accepted: 25 January 2021

Published: 1 February 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 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 (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

On the basis of several forecasts and predictions, the global market for unmanned aerial vehicles (UAVs) is predicted to grow quickly in future years. For instance, the integration of UAVs into the aerospace market of the United States is projected to represent an economic impact of USD 89.1 billion between 2015 and 2025 [1].

A wide variety of actions can be completed by UAVs in different scenarios [2], offering a huge versatility, while artificial intelligence (AI) is more and more frequently employed to enhance their autonomy [3]. Missions incorporate civilian (photography, environmental monitoring, extinguishing wildfires, etc.) and military (reconnaissance, surveillance, target acquisition, etc.) purposes [4–7].

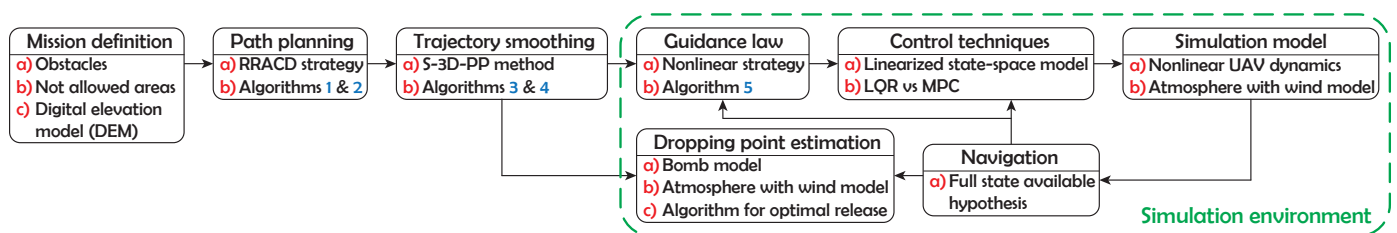
One example of a military mission could be dropping bombs over an objective from an initial location, which is the purpose of the UAV proposed in this article. In this case, the selected aircraft is the F-86 Sabre [8,9], which was hypothetically flown without a pilot for the whole mission. Obviously, other unmanned combat air systems (UCASs) would be more suitable for this mission, such as the X-47B [10,11]. However, it is not easy to find precise information about the aerodynamic characteristics of these combat aircraft, which would allow the implementation of a dynamic nonlinear model for accurate simulations. In contrast, there is detailed information available regarding Sabre, which facilitates the elaboration of those precise nonlinear dynamical simulation models.

To accomplish any autonomous 3D flight, path planning and guidance, navigation, and control methodologies are employed. Path planning determines the flight trajectory

that the UAV should follow to move from an initial point to a final one, avoiding obstacles during the route. By employing sensors like a global positioning system (GPS), inertial units, magnetometers, etc, as well as estimation algorithms like the Extended Kalman Filter [12], the navigation system calculates the actual position and velocity. The proposed trajectory along with the navigation information is computed by the guidance law to determine the main reference signals for the control algorithms. Finally, to ensure the desired motion of a UAV, appropriate control techniques that can determine the actuator signals at each time step are required.

The purpose of this work was to compare two optimal multivariable controllers—in particular, a linear quadratic regulator (LQR) and model predictive control (MPC)—for trajectory tracking in combat missions in which obstacles are averted and a blitz is performed. Hence, both control techniques' behaviors were studied based on flight autonomy, reference following, obstacle avoidance, and bomb impact accuracy. The main goal is to discuss which is the most suitable alternative for this kind of mission using a realistic simulation environment. Additionally, in this comparative study, no landing or takeoff procedures were studied, since the objective consisted of evaluating the flying system's performance during demanding maneuvers.

Figure 1 summarizes the global scheme implemented in this article for the comparative study of the MPC vs. LQR control techniques.



**Figure 1.** Diagram of the technical procedure conducted in the article.

The autonomous flying system implemented in this article is composed of two fundamental blocks: path-planning algorithms and a guidance, navigation, and control (GNC) system.

### 1.1. Path Planning

There is a high diversity of options for implementing successful 3D path planning. Some algorithms are based on sampling fundamentals, with a deep analysis from a 2D perspective [13], but not in a 3D environment; other options are node-based algorithms, which are mathematical structures utilized to model pairwise relationships (structures with vertices and edges), and they have a final purpose of analyzing the cost of examining nodes to find the optimal path. Moreover, bio-inspired algorithms and mathematical models are widely employed, among other techniques [14].

The node-based approach was utilized herein. This type of algorithm presents various methodologies, such as Dijkstra's algorithm [15], A\* [16], D\* [17], or Theta\* [18]. Recently, this technique has been enhanced with modern learning machine methods [14].

Moreover, different planning possibilities have been presented: obstacle avoidance (static and/or dynamic), real-time processing (on- or offline), and optimization parameters (total distance, flight parameters, etc.). The flying system must be able to analyze any obstacle online, although, in order to simplify the problem, an offline static obstacle approach, which attempts to achieve the least demanding trajectory for a UAV, is typically selected.

Many obstacle analysis strategies have been implemented over the years, and have mainly utilized accurate sensors [19] or image processing [20,21]. Nevertheless, in this article, a more simple approach was created, since the obstacles in this approach are defined computationally from the beginning.

Adaptive cell decomposition (ACD) is a strong method for solving physical systems by means of partial differential equations, enabling refined complex 3D Cartesian geometry reconstructions. Consequently, an ACD is proposed in this article, in which the 3D space is decomposed and explored through recursive rewarding cost functions. Afterward, a 3D smooth path-planning algorithm is applied after the ACD in order to obtain a more achievable dynamic trajectory for the UAV during the mission through the introduction of gyration radii in each waypoint from the ACD route. In addition, a reference velocity profile is defined.

## 1.2. Guidance, Navigation, and Control

As mentioned before, GNC systems are applied to ensure appropriate management of any vehicle's movement. First, the guidance function determines the commanded attitude based on the reference trajectory and the navigation information. Then, the control block applies the control actions needed by actuators to fulfill such commands, while stability is guaranteed [22].

In relation to these systems, there are different aspects that play a key role in ensuring the aircraft's proper motion, such as position detection techniques [23] or hardware implementation [24], but only theoretical investigation is proposed in this article. Furthermore, the navigation problem has been simplified by assuming that the state vector of the aircraft (positions, velocities, attitude, etc.) is completely known; in other words, it was defined as completely measurable. Therefore, the navigation system has not been implemented or simulated.

On the other hand, the stability of the aircraft depends on the quality of the control strategy, but also on an adequate performance of the guidance law. In this case, the notable nonlinearity of the system makes the selection of an appropriate methodology a hard task.

### 1.2.1. Control

Nowadays, there is a considerable variety of control methods. The most essential control method is the PID, which enables one to control a system in a closed loop through a proportional, integral, and derivative action [25]. Nevertheless, it is only applicable to single-input/single-output (SISO) systems, so more complex methods are needed to proceed with multiple-input/multiple-output (MIMO) systems. Hence, full state feedback (FSF), in which the dynamics of the process can be changed by modifying the poles of the closed loop system, constitutes a realistic option [26].

The LQR is a suitable control design option that calculates the most adequate feedback gain by diminishing a cost function, which measures changes in the state and control variables [27,28].

Other MIMO control strategies, such as MPC and robust control, can also be applied. The former utilizes the system model and the prediction and control horizon times to predict its future behavior by solving the online optimization algorithm to select the best control action. The MPC can be linear (time-invariant, adaptive, or gain-scheduled) or nonlinear, depending on the recalculation of the plant model and the variation in states and constraints [29,30]. Meanwhile, the latter combines frequency design techniques, such as Bode or Nyquist, and LQR control, and was designed to deal with multiple possible sources of uncertainties, noises, and disturbances [31].

In this article, the LQR and linear time-invariant MPC control methods were selected.

### 1.2.2. Guidance

In guidance laws, two main groups can be differentiated: proportional and those based on predictive control [32]. Proportional guidance was born as an improvement of pure chasing laws [33]; it creates a velocity vector from the aircraft that points to the target (the next desired point on the route), obtaining excessive acceleration. Proportional guidance laws achieve proportional acceleration in accordance with the rotation of the position vector between the target and the UAV.

However, proportional laws present several problems, such as the generation of acceleration instead of velocity, the lack of future prediction, and the inability to introduce constraints or consider perturbations. Thus, optimal control algorithms, such as LQR and MPC, have been explored in recent investigations and implementations to generate predictive control guidance laws, resulting in a more stable tracking of the reference, but also in an increase in the complexity [32,34–37].

Thus, a proportional guidance law, together with the control techniques mentioned above, was chosen to evaluate and study the stability, tracking, robustness against perturbations, and complexity. Additionally, this law contained an improvement: Circular trajectories for the aircraft were imposed so as to avoid the issues of classical proportional laws; hence, it is referred to as a nonlinear guidance law (proportional guidance based on forces) from this point forth.

## 2. Problem Definition

The F-86 Sabre was tasked with a mission to reach a final destination  $q_f$  from an initial point  $q_i$  at a certain velocity and precision so as to accomplish a successful blitz on the objective. The weapons chosen for this purpose were two M117 demolition bombs [38]. Furthermore, several obstacles had to be avoided during the mission.

The aircraft, which actually acted as a UAV, had to conduct autonomous tracking of the reference “dynamic trajectory”, defined as a combination of the path in the 3D Cartesian coordinates  $(x, y, z)$  and the velocity profile that the aircraft must follow, while actuators modified the deflection of the control surfaces and the activation of the throttle lever on the basis of sensor measurements. Only the rudder, ailerons, and elevators were taken into account; hence, the horizontal stabilizer was assumed to be fixed. All of the deflection and throttle lever amplitudes and rates, together with structural limitations and other parameters, were based on [39,40].

The governing equations of the UAV’s motion were given by a typical flight mechanics model and appropriate thrust and fuel consumption equations [41–44], although several fundamental aspects were adapted to this case, such as the time variation in the mass, mass center, and inertia matrix, as well as a non-null atmosphere motion. The UAV model is fully detailed in [45]. A simplified normally distributed wind model was supposed based on [46–48], and average wind velocity and direction values in the operation area were employed (see Table A1 and Equations (A1) and (A2)) [49,50]. The atmospheric parameters were provided by the International Standard Atmosphere (ISA) [43,51,52].

### 2.1. Dynamic Trajectory

In order to accomplish the mission, a 3D representation of the complete environment had to be conducted, in which restrictions were introduced. The flying system had to ensure the construction of a realistic dynamic trajectory for the UAV while avoiding demanding turning rates and excessive accelerations or decelerations as much as possible, since the control and guidance systems were not able to track the reference. To prove the performance of the flying system, a hypothetical military mission is proposed.

#### 2.1.1. Three-Dimensional Space Creation

The whole space in the XY plane was defined by the initial  $q_i$  and final  $q_f$  points. Regarding the z coordinates, the range of values was determined by the sea level altitude and a maximum reference altitude.

Each pair of latitude and longitude coordinates were translated into a couple  $(x, y)$  by means of the great circle distance concept, which defines the shortest length between two points across the globe.



### 2.1.2. Obstacles

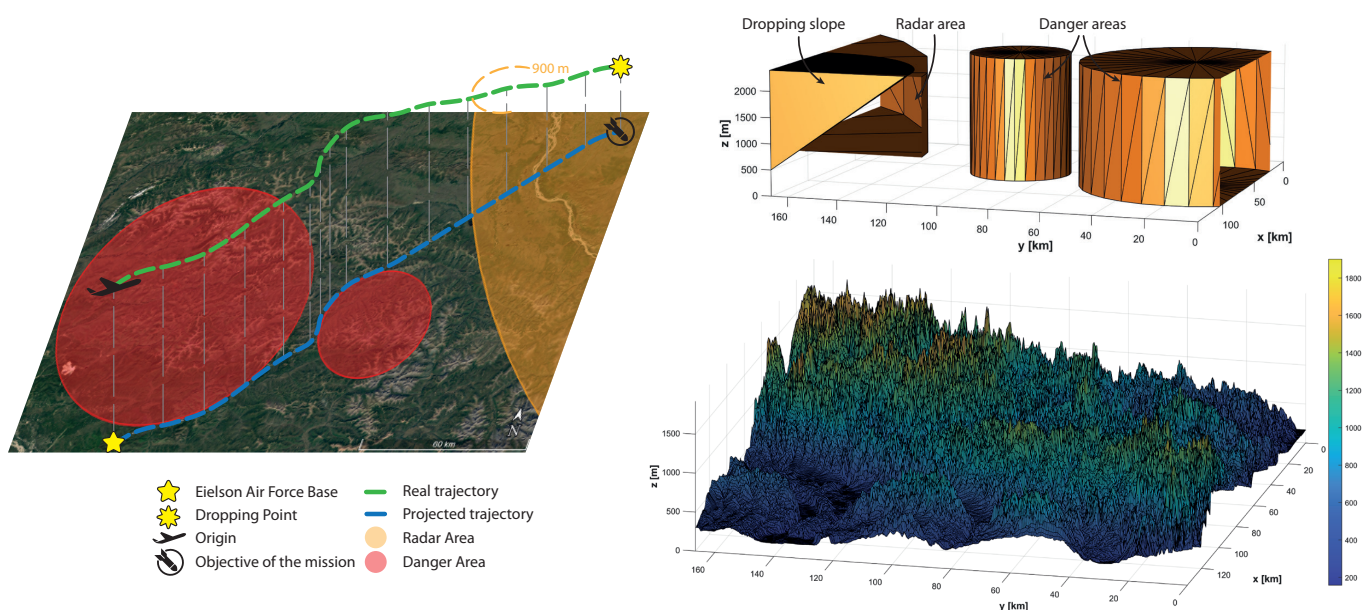
In a real military mission, static and dynamic obstacles can be found; however, only static ones were considered in this case, since no enemy patrols were expected to be encountered. The following elements were used to define the environment of action:

1. Danger area: Zone in which the aircraft was not allowed to enter, generated as a cylinder from the sea level altitude to the maximum altitude.
2. Radar area: Similarly to the danger area, the radar area approximated the scope of an enemy radar. It was defined as a cylinder, with its bottom altitude determined by the minimum altitude at which the UAV could be detected.
3. Dropping slope: This is not a real obstacle, but instead was employed to ensure a correct descent of the aircraft when it approached the objective, generated as an inverted cone whose vertex was  $q_f$ .
4. Ground elevation: Through mapping based on a digital elevation model (DEM) [53] called the National Elevation Dataset (NED) [54], provided by the United States Geological Survey [55], the mountains of the environment were properly defined. The resolution provided by these data was between 30 and 60 m. First, a planar mesh defined by the latitude and longitude coordinates was created in Keyhole Markup Language (KML) [56]; then, it was converted into GPS exchange format (GPX) [57,58], and the elevation data were introduced.

### 2.1.3. Mission Specifications

The F-86 Sabre, which began acting as a flight bomber during the Korean War (1950–1953) [59,60], was hypothetically modernized in order to enhance its military functionality. Nevertheless, this new technology has not yet been tested in this aircraft in autonomous bombing operations; therefore, the United States Air Force (USAF) has decided to conduct a mock blitz in the Center Eastern region of Alaska. The mission will consist of bombarding an uninhabited area close to the border with Canada at a minimum vertical distance of 200 m, starting from the Eielson Air Force Base at an altitude of 1200 m. The Sabre will have to face a radar with a minimum detection altitude of 900 m and two danger areas, with a maximum dropping slope of  $2^\circ$ .

Figure 2 represents a sketch of the operation environment from Google Earth and the required restrictions, which are shown in the 3D space.



**Figure 2.** Initial and final points of the mission and restriction areas (left). Obstacles of the danger and radar areas and the dropping slope (top right), as well as the ground elevation (bottom right).

### 3. Dynamic Trajectory Planning

#### 3.1. Adaptive Cell Decomposition

A typical 3D ACD algorithm attempts to conduct a discretization of the environment in a tree data structure known as Octree. The algorithm verifies a possible collision of each rectangloid with the obstacles at each decomposition level; afterwards, independently of whether a potential collision is identified, all subspaces are divided into eight new children. When the final decomposition level is attained, all children identified as potentially colliding with obstacles are discarded, taking only the collision-free rectangloids. Furthermore, in this case, regardless of whether a subspace is fully included in an obstacle, that subspace is rejected and not divided into new ones at the current decomposition level.

Algorithm 1 shows the pseudo-code utilized in the discretization process. Once the decomposition level ( $n$ ), obstacles, and initial space are defined, the decomposition starts from the first level to the  $n$ th level. At each step, all subspaces of the same level are analyzed (from  $p_0$  to  $p$ ), provided that they are not fully included in any obstacle (complete collision). The term *rectangloid* refers to a 3D matrix that contains all subspaces generated by the algorithm, so that each rectangloid is represented by a position and a group of eight vertices. Then, the vertices of the children are calculated by the *new\_vertices* function and the collision object of each rectangloid is generated by the *collisionBox* function. When the final decomposition is completed, the selection of collision-free rectangloids is accomplished, obtaining a list with their positions in the *rectangloid* matrix.

---

#### Algorithm 1 Adaptive cell decomposition (ACD)

---

```

1:  $n \rightarrow decomposition\_level$ 
2: define Obstacles
3: [rectangloid_size, collision_rectangloid(1)] = initial_rectangloid
4:  $p_0 = p = 1$ 
5: free_rectangloids = []
6: for  $i = 1 : n$  do
7:   for  $s = p_0 : p$  do
8:     if  $s == p_0$  then
9:        $p_0 = p + 1$ ;
10:    end if
11:    [ $-$ , complete_collision] = collision_box_objects(collision_rectangloid(s), obstacles);
12:    if complete_collision == 0 then
13:      [new_vertices] = vertex_generation(rectangloids(:,:s));
14:      for  $j=1:8$  do
15:        rectangloids(:,: ,  $p + j$ ) = new_vertices(j);
16:        [collision_rectangloid( $p + j$ )] = collisionBox( $i$ , rectangloids(:,: ,  $p + j$ ), rectangloid_size);
17:      end for
18:       $p = p + 8$ ;
19:    end if
20:  end for
21: end for
22: for  $s = p_0 : p$  do
23:   [collision,  $-$ ] = collision_box_objects(collision_rectangloids(s), obstacles);
24:   if collision == 0 then
25:     free_rectangloids = [free_rectangloids, s];
26:   end if
27: end for
28: output  $\rightarrow$  planner

```

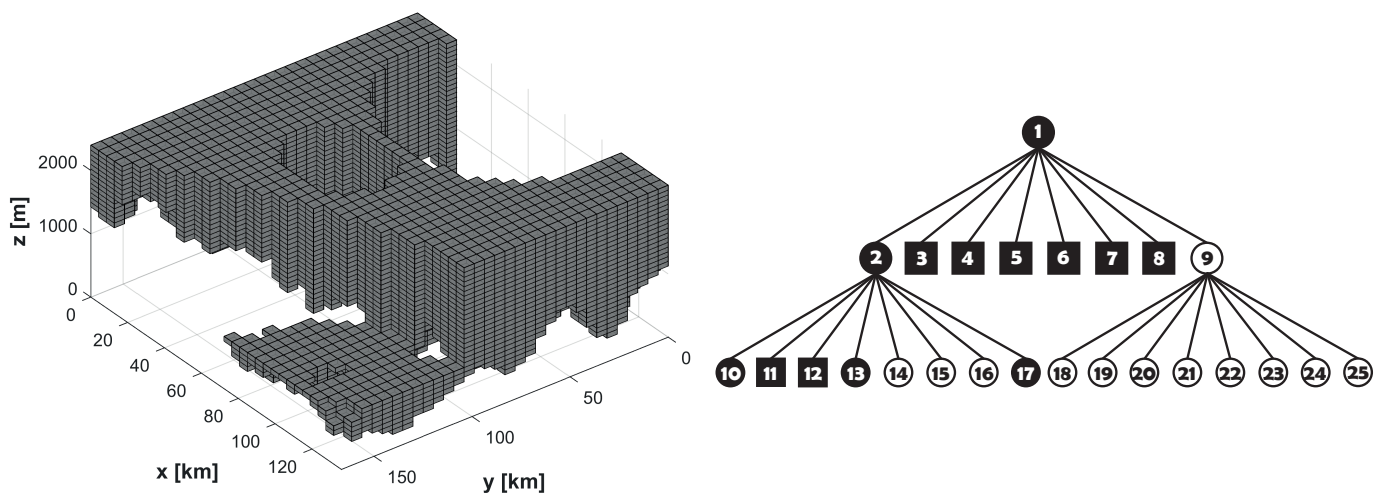
---

This list, together with the vertices of the collision-free subspaces, was employed in the recursive rewarding adaptive cell decomposition (RRACD) algorithm to determine the next possible waypoint on the route.

This methodology is not the most efficient in the computational sense, as it would be more adequate to divide only rectangloids with verified collisions, as is done by the modified ACD algorithm in [14]. However, the basic ACD methodology was preferred, since it facilitates waypoint selection by offering a wider range of possibilities.

In Figure 3, an example of the application of the ACD algorithm to the complete space of study is shown. Moreover, the tree scheme used for the Octree decomposition is represented: The verified and non-verified collisions (black and white circles, respectively) were divided into eight new rectangloids, while the inclusions in the obstacles (black squares) were discarded. This scheme is based on [61].

Furthermore, to diminish the danger of extreme closeness to the terrain, a security distance equivalent to three times the aircraft's wingspan was introduced in the elevations of the ground collision mesh object, allowing for a more secure flight.



**Figure 3.** Total free 3D space calculated using the adaptive cell decomposition (ACD) algorithm with a decomposition level of  $i = 5$  (left) and a tree scheme of a hypothetical Octree block (right).

### 3.2. Recursive Rewarding Adaptive Cell Decomposition Algorithm

An approach to a node-based algorithm was created herein, although the centers of the rectangloids were used instead of edges and vertices, and the optimization parameters comprised the distance and the course and flight path angles. The RRACD algorithm defines a final path  $\rho$  by means of the initial subspaces, so that a starting decomposition of the complete environment is made, assigning to all rectangloids a collision-free grade. The initial level selected for this case was  $i = 5$  ( $n_1$ ), giving, as a result, a number of starting rectangloids of  $n_i = 8^5$ .

Afterward, beginning at the initial point ( $q_i$ ) as a free subspace ( $s_k$ ), the algorithm searches for neighboring rectangloids ( $S_{k+1}$ ) from the initial decomposition and divides them into new ones ( $s_{k+1}$ ) by utilizing the ACD code at a level of  $n_2$ , which is equal to 3 for this operation. It is important to mention that the complete decomposition  $n = n_1 + n_2$  is limited, since the final rectangloid dimensions must be greater than the planar accuracy of the ground elevation mesh. The most adequate neighboring option is found by the sum of  $\mathcal{D}_1$  and  $\mathcal{D}_2$ , two rewarding functions that measure the cost of selecting a future destination. This procedure is repeated until the final point ( $q_f$ ) is attained.

$\mathcal{D}_1$  is obtained from the path analysis of  $s_k$  to  $s_{k+1}$  ( $s_k \rightarrow s_{k+1} \Rightarrow \mathcal{D}_1$ ) and  $\mathcal{D}_2$  from  $s_{k+1}$  to  $q_f$  ( $s_{k+1} \rightarrow q_f \Rightarrow \mathcal{D}_2$ ). Each one is defined by a set of  $R_{m,n}$  partial functions in combination with a Gaussian function  $g(R_{m,n})$  and the weight parameters  $\eta_1$  (for  $\mathcal{D}_1$ ) and  $\eta_2$  (for  $\mathcal{D}_2$ ).

The set  $R_{m,n}$  is a group of  $m = 1 \dots N, n = 1 \dots N$  partial functions that involve 3D flying characteristics. In the algorithm, a total of  $N = 3$  functions are defined (position  $n$ ) and further divided into types 1 and 2 (position  $m$ ) in relation to the subpaths  $s_k \rightarrow s_{k+1}$  and  $s_{k+1} \rightarrow q_f$ , respectively. Type 1 functions take into account previous course and flight path angles to measure their variations, while type 2 functions only consider the angles of the subpath.

1. Distance: Given by the expressions below, where  $M_{distance}(s_i \rightarrow s_j)$  is the Euclidean distance between the chosen subspaces  $s_i$  and  $s_j$ , and  $M_{direct}$  is the straight line between  $q_i$  and  $q_f$ .

$$R_{11}(i, j) = R_{21}(i, j) = \frac{M_{distance}(s_i \rightarrow s_j)}{M_{direct}} \in \mathbb{R} : [0, 1] \quad (1)$$

$$M_{distance}(s_i \rightarrow s_j) = \sqrt{(s_{ix} - s_{jx})^2 + (s_{iy} - s_{jy})^2 + (s_{iz} - s_{jz})^2} \quad (2)$$

2. Flight path angle: Where  $\gamma_0$  is the flight path angle of the previous branch of the path and  $\gamma$  is one of the vectors given by  $s_i \rightarrow s_j$ . The functions must provide a value between  $-1$  and  $1$ , so that in cases in which  $|\gamma - \gamma_0|, |\gamma| > 45^\circ$ , the output will be  $1$ .

$$R_{12}(i, j) = \tan(\gamma - \gamma_0), R_{22}(i, j) = \tan(\gamma) \in \mathbb{R} : [-1, 1] \quad (3)$$

$$\gamma = \arctan\left(\frac{s_{jz} - s_{iz}}{\sqrt{(s_{jx} - s_{ix})^2 + (s_{jy} - s_{iy})^2}}\right) \quad (4)$$

3. Course angle:  $\Psi_0$  is the course angle of the previous branch of the path and  $\Psi$  is one of the vectors given by  $s_i \rightarrow s_j$ . As before, when  $|\Psi - \Psi_0|, |\Psi| > 45^\circ$ , the output in the code will be  $1$ .

$$R_{13}(i, j) = \tan(\Psi - \Psi_0), R_{23}(i, j) = \tan(\Psi) \in \mathbb{R} : [-1, 1] \quad (5)$$

$$\Psi = \arctan\left(\frac{s_{jy} - s_{iy}}{s_{jx} - s_{ix}}\right) \quad (6)$$

Regarding the Gaussian function  $g(R_{m,n})$  utilized to analyze the reward in executing a possible action, it is defined by the expression (7), in which the transition cost values ( $s_k \rightarrow s_{k+1}, s_{k+1} \rightarrow q_f$ ) have been normalized within the boundaries  $[0, 1]$ .

$$g(R_{m,n}) = \frac{\sin(\pi \cdot R_{m,n} + \pi/2) + 1}{2} \quad (7)$$

It is important to note that the higher the effort  $R_{m,n}$ , the smaller the reward  $g(R_{m,n})$ , and vice versa. All of these rewards are collected in the vector  $G_m(i, j)$ :

$$G_m(i, j) = [g(R_{m,1}(i, j)), g(R_{m,2}(i, j)), \dots, g(R_{m,N}(i, j))]. \quad (8)$$

Finally,  $G_m(i, j)$ , the priority parameters ( $\eta_1, \eta_2$ ), and a new parameter  $\xi$  are employed to build the a final rewarding function  $\mathcal{D}$ , which is composed of  $\mathcal{D}_1$  and  $\mathcal{D}_2$ . The neighbor with the greatest reward is selected as the next waypoint on the route. The reward of selecting a subspace  $s_j$  is given by the following expressions:

$$\mathcal{D}_1 = \eta_1 \cdot G_1(s_k, s_j) + \xi_{s_j}, \quad \eta_1 = [\eta_{1D}, \eta_{1\gamma}, \eta_{1\Psi}] \quad (9)$$

$$\mathcal{D}_2 = \eta_2 \cdot G_2(s_j, q_f), \quad \eta_2 = [\eta_{2D}, \eta_{2\gamma}, \eta_{2\Psi}] \quad (10)$$

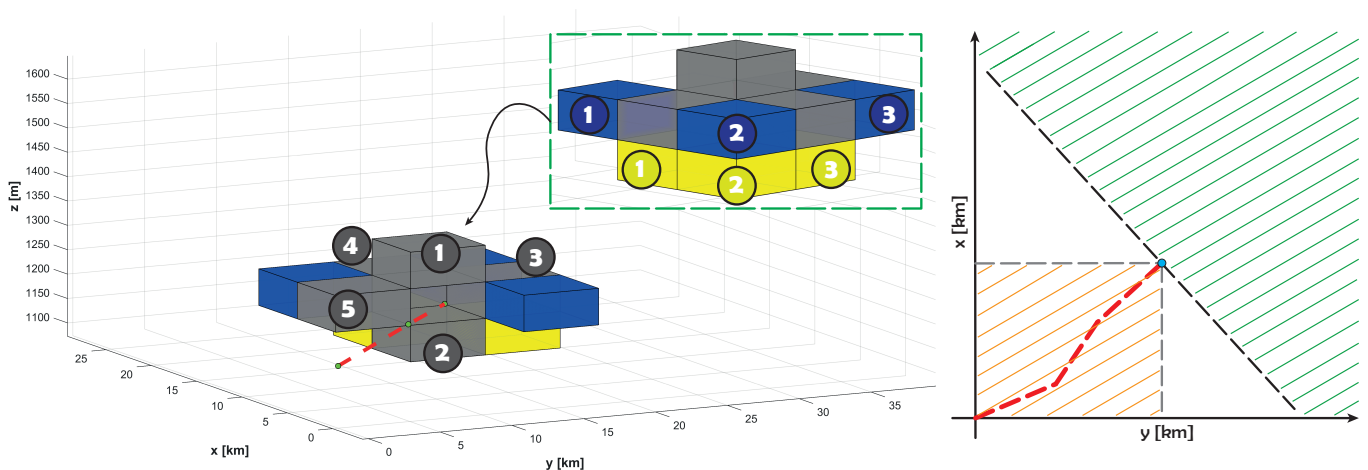
The parameter  $\xi$  is a variable reward value that is assigned to the neighbors  $s_{k+1}$  that are considered as very problematic or extremely necessary given the aircraft position and the obstacle locations. For instance, before entering into the radar area, the aircraft must



decrease its altitude significantly to cross it successfully, so that only the neighbors that are located in a lower position at each step will be scored with a positive  $\xi$ .

Regarding the neighborhood choice, the block subspaces  $s_{k+1}$  located on the top, bottom, sides, and in front (gray) of the subspace  $s_k$  are evaluated as neighbors (see Figure 4). Furthermore, three diagonal rectangloids (blue) are permanently available neighbors for improving the algorithm's capabilities in the path-building process, while the other three on the bottom front, diagonal, and right sides (yellow) are introduced when the aircraft is close to the radar's perimeter in order to facilitate the descent. If the UAV has already been to one of these possible subspaces, it is automatically eliminated from the list of destinations. Once the neighborhood has been obtained, the ACD algorithm is applied at each subspace, and afterward, the centers of all new rectangloids for which the rewarding function is used are calculated so as to choose the next most appropriate destination.

In addition, dynamic restrictions are applied after all centers have been obtained. The centers that suppose a change in the course angle greater than  $90^\circ$  or imply the entrance of the UAV into a fully forbidden zone (see Figure 4) are discarded. Moreover, if the selection of a center entails the UAV navigating a trajectory with a flight path angle  $|\gamma| > 5^\circ$ , this center is also rejected.



**Figure 4.** Examples of the possible neighbor subspaces at a point in the path (left); the available area in the XY plane in green and the fully forbidden zone in orange (right).

In Algorithm 2, the RRACD process is shown. First, the initial and local decomposition levels are defined, together with the rest of the variables. Then, the subspaces that have already been visited are discarded and the neighborhood is calculated. The ACD code is applied to each rectangloid and the available centers are calculated by the function *center\_selection*; in the case that there are no centers, a new process with a higher level of decomposition is performed. The rewards of the centers are obtained, selecting the point with the greatest reward.  $M_{distance}$  is calculated every time, so that if the last waypoint is fairly close to  $q_f$ , the algorithm takes it directly.

The final reference trajectory, which is shown in Figure 5, offers acceptable flight conditions and avoids all imposed obstacles. However, this is not the best possible solution, and different combinations of weighting parameters should be selected to enhance the results.

**Algorithm 2** Recursive rewarding adaptive cell decomposition (RRACD)

---

```

1: define  $q_i, q_f, obstacles, \eta_1, \eta_2, d_{min}, \xi_{areas}$ 
2:  $n_1 \rightarrow$  initial decomposition level
3:  $n_2 \rightarrow$  local decomposition level
4:  $[rectangloids, free\_rectangloids] = ACD(initial\_rectangloid, n_1, [])$ 
5:  $wp = q_i$ 
6:  $M_{distance} = Euclidean\_distance(q_i, q_f)$ 
7:  $[s_k] = [s_{k_i}] = cube\_finding(q_i, rectangloids, free\_rectangloids)$ 
8:  $[\Psi] = initial\_angle(q_i, q_f)$ 
9: while  $M_{distance} > d_{min}$  do
10:   if  $s_k == s_{k_i}$  then
11:      $free\_rectangloids(find(free\_rectangloids == s_k)) = []$ ;
12:   else
13:     for  $j = 1 : length(s_{k+1})$  do
14:        $free\_rectangloids(find(free\_rectangloids == s_{k+1}(j))) = []$ ;
15:     end for
16:   end if
17:    $s_{k+1} = neighborhood(s_k, rectangloids, free\_rectangloids, wp(end, :), \vec{r}_{radar}, R_{radar})$ ;
18:    $M_{distance} = Euclidean\_distance(q_i, q_f)$ ;
19:    $local\_divisions = n_2$ ;
20:    $[centers] = []$ ;
21:   while  $length(centers) == 0$  do
22:     for  $i = s_{k+1}$  do
23:        $[rectangloids', free\_rectangloids'] = ACD(rectangloids(:, :)$ 
24:          $, s_{k+1}), local\_divisions, obstacles)$ ;
25:        $[centers] = center\_selection(rectangloids', free\_rectangloids')$ ;
26:     end for
27:      $local\_divisions = local\_divisions + 1$ ;
28:   end while
29:    $[center\_chosen, s_k, \Psi] = rewards(wp(end, :), centers, q_f, \xi_{areas}, \Psi(end))$ ;
30:    $wp = [wp; center\_chosen]$ ;
31: end while
32: return  $Final\_path\_waypoints \rightarrow wp$ 

```

---

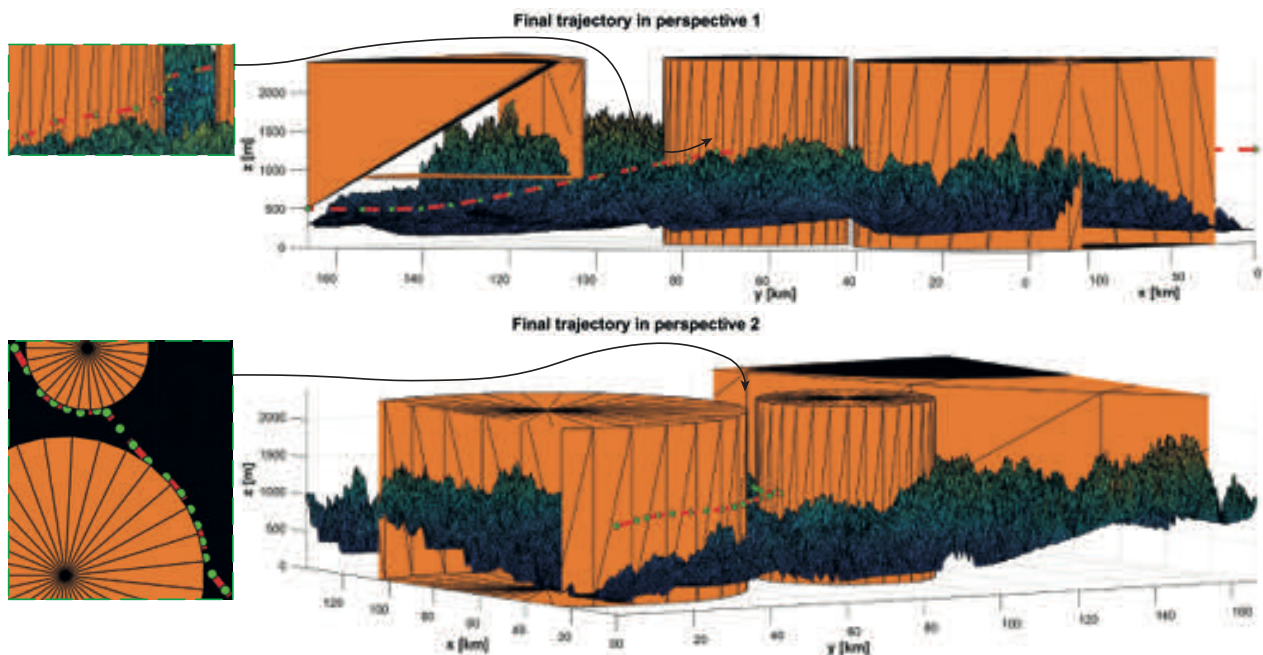


Figure 5. Final trajectory obtained by means of the recursive rewarding adaptive cell decomposition (RRACD) algorithm.

### 3.3. Trajectory Smoothing

Although the path obtained by the RRACD algorithm is appropriate for UAV guidance, sudden changes in direction when passing a waypoint are impossible for an aircraft to achieve, which could thus provoke excessive maneuvers by the UAV in the guidance process; therefore, the radii of gyration are applied to obtain a smooth trajectory, which is easier to track than the previous one. However, this smooth path could collide with obstacles; hence, the possibility of collisions is analyzed to ensure that this does not happen, and if it does occur, the radii of gyration are diminished in the conflict regions. Thus, the smooth 3D path-planning algorithm finds the optimal trajectory in terms of low angle variation rates and obstacle avoidance based on some concepts from [62]. Moreover, a velocity profile is designed by the algorithm.

The algorithm is applied to each non-aligned group of three waypoints. In Figure 6, an example of the application of the radii of gyration is represented, in which points  $s_1$  and  $s_2$  are the tangential points among the arc and both lines, vectors  $\vec{v}_1$ ,  $\vec{v}_2$ ,  $\vec{r}_1$ , and  $\vec{r}_2$  are the tools utilized to define points  $s_1$  and  $s_2$ , as well the arc center ( $O$ ), and  $R$  is the radius. The directions of vectors  $\vec{r}_1$  and  $\vec{r}_2$  are the result of the following vectorial product:  $\vec{r}_i = \vec{v}_i \times (\vec{v}_1 \times \vec{v}_2)$ .

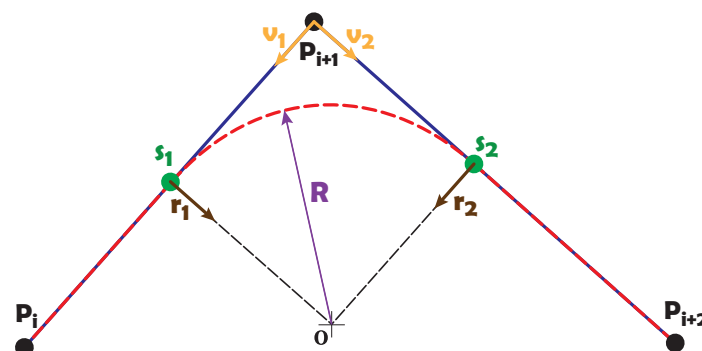


Figure 6. Radii of gyration: reference trajectory (blue), waypoints ( $P$ ), smooth path (red), radius ( $R$ ), center ( $O$ ), and tangential points ( $s$ ).

The generation of the final smooth trajectory is made by means of two steps at each iteration: first, the path and velocity profile using Algorithm 3, and second, analysis of the errors in the trajectory building and collision of the new path with the obstacles, which are conducted using Algorithm 4.

---

**Algorithm 3** Smooth 3D path simple design (S-3D-PSD)
 

---

```

1: define  $path, \gamma, \Psi, V_{min}, V_{max}, V_0, \frac{dV}{dS}, ds, n_0, n_{max}, factors$ 
2:  $pathref = [path(1,:) V_0]$ ;
3:  $S = [0]$ ;
4:  $s_v = []$ 
5:  $i = 1$ ;
6:  $flag = 0$ ;
7: while not ( $i == size(path, 1)$ ) do
8:   if not ( $(\Psi(i + 1) - \Psi(i)) == 0$ ) or not ( $(\gamma(i + 1) - \gamma(i)) == 0$ ) then
9:      $[\vec{v}_1, \vec{v}_2] = vector\_v(path(i : i + 2, :))$ ;
10:     $\vec{v}_{12} = \vec{v}_1 \times \vec{v}_2$ ;
11:     $[\alpha] = angle(\vec{v}_1, \vec{v}_2)$ ;
12:     $n = load\_factor(n_0, factors(i), \Psi)$ ;
13:     $R = \max((pathref(end, 4))^2 / (9.81 \cdot \sqrt{n^2 - 1}), (pathref(end, 4))^2 / (9.81 \cdot \sqrt{n_{max}^2 - 1})$ ;
14:     $[s] = s\_distance(R, \alpha)$ 
15:     $s_v = [s_v s; i]$ ;
16:     $\vec{r}_1 = \vec{v}_{12} \times \vec{v}_1$ ;
17:     $\vec{r}_2 = \vec{v}_2 \times \vec{v}_{12}$ ;
18:     $[s_1, s_2] = s\_point(\vec{v}_1, \vec{v}_2, s, path, \Psi)$ ;
19:     $O = center(s_1, s_2, r_1, r_2, v_1, v_2)$ ;
20:     $pc = circle\_points(v_{12}, s_1, s_2, O, R, ds)$ ;
21:     $[points_{ref}, S_{tot}, i_{val}, flag_{val}] = build\_path1(V_{min}, V_{max}, \frac{dV}{dS}, pathref(end, :), path(i : i + 2, :), \Psi(i : i + 3), \gamma(i : i + 3), v_1, v_2, s_1, s_2, ds, flag, pc)$ ;
22:  else
23:     $[\vec{v}_1] = vector\_v(path(i : i + 1, :))$ ;
24:     $[points_{ref}, S_{tot}, i_{val}, flag_{val}] = build\_path2(V_{min}, V_{max}, \frac{dV}{dS}, pathref(end, :), path(i : i + 2, :), \Psi(i : i + 3), \gamma(i : i + 3), v_1, ds, flag)$ ;
25:  end if
26:   $pathref = [pathref; points_{ref}]$ ;
27:   $S = [S S_{tot}]$ ;
28:   $flag = flag_{val}$ ;
29:   $i = i_{val}$ ;
30: end while

```

---

**Algorithm 4** Smooth 3D path planning (S-3D-PP)

---

```

1: define  $path, \gamma, \Psi, V_{min}, V_{max}, V_0, \frac{dV}{dS}, ds, n_0, n_{max}, precision, obstacles$ 
2:  $factors = zeros(size(path, 1) - 1)$ 
3:  $solution = 0$ 
4: repeat
5:    $[pathref, S, s_v] = S\_3D\_PSD(path, \gamma, \Psi, V_{min}, V_{max}, V_0, \frac{dV}{dS}, ds, n_0, n_{max}, factors);$ 
6:    $[error, locations_{error}] = error\_finding(pathref, s_v);$ 
7:    $[collision, locations_{collision}] = collision\_finding(pathref, obstacles);$ 
8:   if  $error == true$  then
9:      $[factors_{new}] = modification\_factors(factors, locations_{error}, precision);$ 
10:  else if  $collision == true$  then
11:     $[factors_{new}] = modification\_factors(factors, locations_{collision}, precision);$ 
12:  else if  $(error \text{ and } collision) == false$  then
13:     $solution == 1$ 
14:  end if
15:   $factors = factors_{new};$ 
16: until  $solution == 1$ 
17:  $[\Psi_{new}, \gamma_{new}, \frac{d\Psi}{dt}] = new\_angles(pathref);$ 
18: return  $Final\_trajectory\_parameters \rightarrow wp = [pathref \ \Psi_{new} \ \gamma_{new} \ \frac{d\Psi}{dt}]$ 

```

---

Regarding Algorithm 3, the first step consists of initializing the required variables: The path, yaw, and flight path angles obtained from the RRACD code; the maximum, minimum, and initial speeds in the velocity profile; the maximum positive velocity variation per meter ( $\frac{dV}{dS}$ ); the precision of the trajectory building ( $ds$ ); the minimum load factor used at each curve ( $n_0$ ); the maximum allowed load factor ( $n_{max} = 5.5$ ) [40]; the vector  $factors$ . The vector  $factors$  contains the gradient of the load factor with respect to the yaw angle change ( $\frac{dn}{d\Psi}$ ), so that when a change in  $\Psi$  takes place, an additional load factor is applied. Therefore, the formula for the load factor in the case of  $i$  is given by the expression:

$$n_i = n_0 + factors(i) \cdot |\Psi_{i+1} - \Psi_i| / 90^\circ. \quad (12)$$

Afterward, the code examines the complete reference path by applying the radii of gyration when any change in the course or flight path angles occurs, which can be approximated by  $R = V^2 / (g \cdot \sqrt{n^2 - 1})$  [43]. Vectors  $\vec{v}_1$  and  $\vec{v}_2$  and their angle  $\alpha$  are obtained; then, the load factor is obtained from the expression (12) to calculate the radii of gyration, which are restricted to the radius corresponding to  $n_{max}$ . After that, the distance  $|s_i - P_{i+1}|$  and the vectors  $\vec{r}_1$  and  $\vec{r}_2$  are calculated, from which points  $s_1$  and  $s_2$  and the center  $O$  are found.

Finally, the points from  $P_i$  to  $P_{i+2}$ , which contain the arc, are obtained by the  $build\_path1$  function. These points also include all of the velocities as a fourth element, except the 3D coordinates. The  $flag$  variable can be 0 or 1 and determines whether a radius of gyration has been applied in the previous group of waypoints in order for the path to be built properly. In the case of an aligned group of waypoints, the  $build\_path2$  function generates a straight line.

Once a first dynamic trajectory is generated, Algorithm 4 attempts to find possible errors in the construction due to overlapping arcs and studies whether the new trajectory collides or not. At the beginning, the load factors are set at low values to create big radii of gyration in order for the algorithm to be able to find the optimal solution by



means of optimizing the vector *factors*. Therefore, the code analyzes which waypoints are problematic and adds to their respective *factors* parameters and *precision* values, increasing the load factor in the points that have excessive radii of gyration (these positions are defined by the vector *locations*) and trying to prevent the arcs from overlapping or colliding with obstacles. Every time a building error or collision is found, the whole code generates a new trajectory with the improved *factors<sub>new</sub>* vector; hence, only when the new path is perfectly well defined can the algorithm end and the solution be obtained.

Finally, a new trajectory course and its time derivative and flight path angles are calculated with the *new\_angles* function.

In Figure 7, the final results of the algorithms are shown. The parameters utilized in this case are:

$$\begin{cases} V_{min} = 190 \frac{m}{s} \\ V_{max} = 240 \frac{m}{s} \end{cases} \quad \begin{cases} V_0 = 200 \frac{m}{s} \\ \frac{dV}{dS} = 4 \frac{m/s}{km} \end{cases} \quad \begin{cases} ds = 20 m \\ n_0 = 1.075 \end{cases} \quad \begin{cases} n_{max} = 5.50 \\ precision = 1 \times 10^{-2} \end{cases} \quad (13)$$

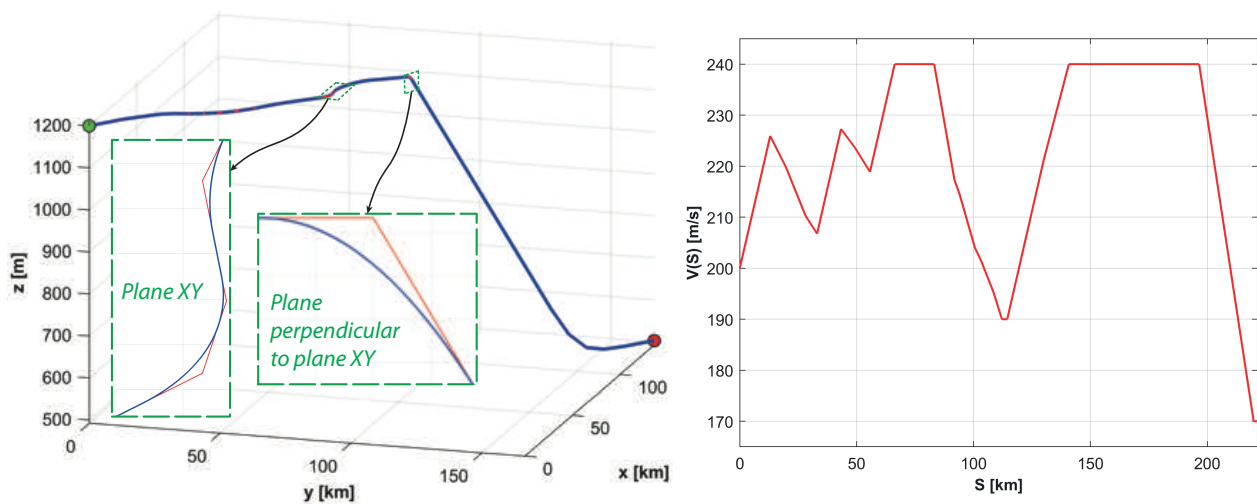


Figure 7. Final trajectory obtained by the 3D smooth path-planning algorithm (left, original in red and final in blue) and the velocity profile as a function of the curvilinear distance *S* (right).

The velocity profile is created with the purpose of reducing the speed during curves and descents, while it is increased for straight lines in order to minimize the flight time and to make the dynamic trajectory easier to follow. In addition, the desired velocity of a bomb dropping at  $V = 170$  m/s is selected to accurately impact the objective; hence, the UAV must progressively diminish its speed when it is close to said objective.

#### 4. Guidance Law

A nonlinear guidance law induces an acceleration in an aircraft to incorporate it into the reference trajectory. In this case, the acceleration is only in the horizontal plane, seeing that the control is conducted on the altitude state variable and not over the flight path angle.

This force is applied perpendicularly to the velocity vector ( $\vec{V}$ ) according to the expression (14).  $N$  is a proportional gain that must be adjusted to achieve greater or lower accelerations depending on the case because an excessive value can produce an unstable flight with many oscillations, while a low value generates poor adaptive guidance when the UAV faces changes in the reference.  $\vec{r}$  is the relative position vector and  $\eta$  is the angle between  $\vec{V}$  and  $\vec{r}$ .

$$a_{horizontal} = N \frac{V^2}{r} \sin(\eta) \quad (14)$$

The desired trajectory of interception is circular, so that the length of  $\vec{r}$  is defined by the radius  $R$  of such a circular motion. Both the target in the reference path and the

instantaneous aircraft location are points of this circumference. In Figure 8, this process can be observed.

In addition, in Figure 8, a simplification of the GNC system scheme is shown. The code composed of Algorithms 1–4 sends position and velocity data to the guidance system. The guidance law creates an utter reference in altitude, attitude (roll and yaw angles only), and velocity based on the navigation information. These new references are analyzed by the control system, which generates the required control action signals. Note that, as mentioned in Section 1.2, the navigation problem has been simplified by assuming that the state vector of the aircraft is completely known.

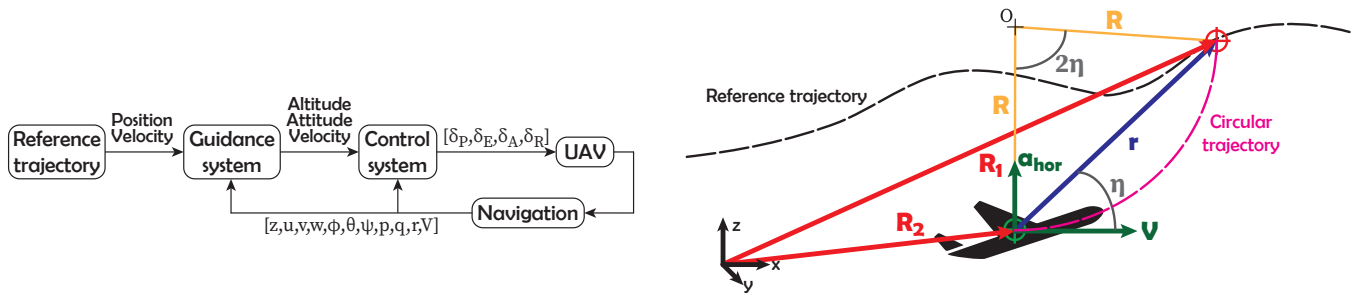


Figure 8. Conceptual guidance system (left) and nonlinear guidance law (right) schemes.

The attitude control cannot accept accelerations; consequently, the horizontal force is translated into a desired course angle ( $\Psi$ ), from which the required yaw angle ( $\psi$ ) that the aircraft must achieve is obtained. At each sampling time ( $T_s$ ), a command of the course angle change ( $\Delta\Psi$ ) is given. The expression of this angle change is as follows:

$$\Delta\Psi \approx \frac{a_{horizontal}}{V} T_s = N \frac{V}{r} \sin(\eta) T_s. \tag{15}$$

The velocity and bank angle ( $\eta$ ) are assumed as constants and the flight path angle ( $\gamma$ ) as null, and a symmetrical flight is also supposed, resulting in the bank and roll angles ( $\phi$ ) being equal [32]. These hypotheses, defined by the expressions (16) of uniform circular motion in the horizontal plane, are no longer true if the sideslip and flight path angles are very different from zero, but this is the best approximation for achieving a set of equations that allows one to obtain a manageable solution of the reference roll angle.

$$T = D, \quad \frac{W}{g} \frac{V^2}{R} = L \sin(\eta) = L \sin(\phi), \quad W = L \cos(\eta) = L \cos(\phi) \tag{16}$$

Hence, the new horizontal acceleration is  $\frac{V^2}{R}$ . These conditions are translated into expressions for the load factor and the roll angle of the aircraft, defined by the following equations:

$$W = L \cos(\phi) \Rightarrow n = \frac{L}{W} = \frac{1}{\cos(\phi)} \Rightarrow \phi = \arccos\left(\frac{1}{n}\right) \tag{17}$$

$$\frac{W}{g} \frac{V^2}{R} = L \sin(\phi) \Rightarrow n = \sqrt{\left(\frac{V^2}{gR}\right)^2 + 1} \Rightarrow n = \sqrt{\left(\frac{2V^2 \sin(\eta)}{gr}\right)^2 + 1}. \tag{18}$$

In addition, the load factor must be between boundaries in order to guarantee stall avoidance and exceeding the structural limits; therefore, the load factor value must ensure that:

$$n \leq \min(n_{max} = 5.5, \frac{\rho V^2 S_w C_{L_{max}}}{2W}). \tag{19}$$

Once the fundamentals of this guidance law are detailed, Algorithm 5 is implemented, which attempts to find the desired target point at each time step and defines the reference in the velocity, altitude, and roll and yaw angles.

**Algorithm 5** Nonlinear guidance law

---

```

1: define waypoints,  $T_s, x, y, z, V, \Psi, u, v, w, \phi, \theta, n_{max}, S_w, m, CL_{max}, threshold$ 
2:  $R = assign\_radius(V)$ 
3:  $[index_1] = next\_point(waypoints, R, x, y)$ 
4:  $x_{ref} = waypoints(index_1, 1)$ 
5:  $y_{ref} = waypoints(index_1, 2)$ 
6:  $[index_2] = closest\_point(waypoints, x, y)$ 
7:  $z_{ref} = waypoints(index_2, 3)$ 
8:  $V_{ref} = waypoints(index_2, 4)$ 
9:  $\vec{V}_0 = [V \cos(\Psi) \ V \sin(\Psi)]$ 
10:  $\vec{r} = [x_{ref} - x \ y_{ref} - y]$ 
11:  $[\eta] = angle(\vec{V}_0, \vec{r})$ 
12: if  $\eta == 0$  then
13:    $\Delta\Psi = 0$ 
14: else
15:    $\Delta\Psi = sign(\eta)(2VT_s / |\vec{r}|)$ 
16: end if
17:  $\Psi = \Psi + \Delta\Psi$ 
18:  $n = \min(n_{max}, \sqrt{((2V^2 \sin(\eta)) / (g|\vec{r}|))^2 + 1}, (\rho V^2 S_w CL_{max}) / (2W))$ 
19:  $\phi_{ref} = sign(\eta) \arccos(1/n)$ 
20:  $[\Psi R_B] = rotation\_matrix(\phi, \theta)$ 
21:  $\vec{V} = [u; v; w]$ 
22:  $\vec{V}_\Psi = \Psi R_B \cdot \vec{V}$ 
23:  $\psi = \Psi - \arctan(\vec{V}_\Psi(2, 1) / \vec{V}_\Psi(1, 1))$ 
24: if  $|[x, y, z] - waypoints(end, 1 : 3)| < threshold$  then
25:    $stop = 1$ 
26: else
27:    $stop = 0$ 
28: end if

```

---

First, the current flight parameters are defined, together with the aircraft parameters ( $S_w$ ,  $m$ , and  $CL_{max}$ ) and the threshold, which is the length that specifies whether the simulation stops when the distance between the aircraft and the final destination is lower. *waypoints* is provided by the dynamic trajectory-planning process and includes reference 3D coordinates and velocities.

The radius  $R$  is calculated from the velocity, taking greater values for high velocities and lower values if the speed is minimum. Afterward, the coordinates  $(x, y)$  of the intersection point between the circle, whose radius is  $R$  and whose center is the aircraft position, and the trajectory are obtained. There are two intersections, so the one with a higher *waypoints* index is chosen. Second, repeating the same procedure but without the radius, the closest point of the reference path to the aircraft position is selected to specify the altitude and velocity to be tracked.

Afterward, the relative position and velocity vectors in the  $XY$  plane are obtained, so that the changes in the course angle ( $\Delta\Psi$ ), the load factor ( $n$ ), and the reference roll angle ( $\phi$ )

can be calculated by means of the previously explained equations. Then, the reference yaw angle is achieved by means of the rotation matrix  ${}^{\Psi}R_B$ , which translates the velocity in body axes into XY plane motion with the aircraft's direction, as well as the course angle. Finally, the distance from the aircraft to the final destination is analyzed, so that if it is lower than the threshold, the simulation ends.

## 5. Control Strategies

In order to develop a control system, the mathematical model must be linearized beforehand with respect to an equilibrium point. The equilibrium conditions correspond to the trimmed point of the aircraft in a vertical planar motion characterized by the given velocity and altitude values. In this case,  $V_{trim} = 215$  m/s and  $z_{trim} = 848.4$  m are utilized, which are the mean values of the dynamic trajectory obtained in the previous chapter.

If the system of equations obtained from the linearization process is rearranged, a linearized state–space model can be achieved, given by the expression (20), in which  $\mathbf{x}$  represents the state variables,  $\mathbf{u}$  the control variables, and  $\mathbf{d}$  the modeled perturbations.  $A$  is the system matrix and  $B$  is the control matrix, while  $A_d$  is the disturbance matrix. All variables are given in increments with respect to their corresponding equilibrium position values. Attending to the Equations (A3) and (A4) of these matrices, all variables are interrelated; hence, the longitudinal and lateral direction motions are coupled.

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + A_d\mathbf{d} \rightarrow \Delta\dot{\mathbf{x}} = A \cdot \Delta\mathbf{x} + B \cdot \Delta\mathbf{u} + A_d \cdot \mathbf{d} = A \cdot \Delta\mathbf{x} + \bar{B}[\Delta\mathbf{u} \quad \mathbf{d}] \quad (20)$$

The chosen state variables are the body velocities ( $u$ ,  $v$ , and  $w$ ) and the angular body velocities ( $p$ ,  $q$ , and  $r$ ), the Euler angles ( $\phi$ ,  $\theta$ , and  $\psi$ ), the altitude ( $z$ ), and the total body velocity ( $V$ ). The control variables are the throttle lever activation ( $\delta_p$ ) and elevator ( $\delta_E$ ), the ailerons ( $\delta_A$ ), and the rudder deflections ( $\delta_R$ ). To facilitate the control system design, the perturbation variables are also added to the control variables, i.e., the wind velocities in the fixed axes ( $V_{x_w}$ ,  $V_{y_w}$ , and  $V_{z_w}$ ) and the drag compressibility coefficient ( $CD_{wave}$ ), since they are zero when the Mach number is below the divergence number ( $M_{div} = 0.85$ ) [43]. The implemented linearized state–space model is as follows:

$$\Delta\mathbf{x} = [ \Delta z \quad \Delta u \quad \Delta v \quad \Delta w \quad \Delta\phi \quad \Delta\theta \quad \Delta\psi \quad \Delta p \quad \Delta q \quad \Delta r \quad \Delta V ]^T \quad (21)$$

$$\bar{\Delta\mathbf{u}} = [\Delta\delta_p \quad \Delta\delta_E \quad \Delta\delta_A \quad \Delta\delta_R \quad V_{x_w} \quad V_{y_w} \quad V_{z_w} \quad CD_{wave}]^T \quad (22)$$

$$\begin{cases} \Delta\dot{\mathbf{x}}_{11 \times 1} = A_{11 \times 11} \cdot \Delta\mathbf{x}_{11 \times 1} + \bar{B}_{11 \times 8} \cdot \bar{\Delta\mathbf{u}}_{8 \times 1} \\ \mathbf{y}_{11 \times 1} = C_{11 \times 11} \cdot \Delta\mathbf{x}_{11 \times 1} + D_{11 \times 8} \cdot \bar{\Delta\mathbf{u}}_{8 \times 1} \end{cases}, \quad \begin{cases} C_{11 \times 11} = I_{11 \times 11} \\ D_{11 \times 8} = [0]_{11 \times 8} \end{cases} \quad (23)$$

In addition, regarding the sample time, it must fulfill the Nyquist theorem [63], in which this parameter must be less than the settling time of the system to allow an acceptable control. In this case, the settling time of the system is related to the sixth eigenvalue of the system matrix. The selected sample time is  $T_s = 0.021$  s, so that it satisfies the requirement:

$$T_s = 0.021 \text{ s} < \frac{1}{2 \cdot \text{Re}(\lambda_6)} = 0.18 \text{ s}. \quad (24)$$

### 5.1. Linear Quadratic Regulator

An LQR is an optimal control technique that provides the best possible performance with respect to some given performance index. The LQR design problem involves the design of a state feedback controller  $K$  such that the objective function  $\mathcal{J}$  is minimized [64]. In this technique, a feedback gain matrix is designed that minimizes the objective function in order to achieve some compromise between the use of control effort, the magnitude, and the speed of response that guarantee a stable system. The  $\mathcal{J}$  function proposed in

this work is the well-known cost index described in (25) [27]; although the  $N_{ru}$  weighting matrix is typically considered null, the cost function becomes a quadratic one.

$$\mathcal{J} = \int_0^{\infty} (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + 2\mathbf{x}^T N_{ru} \mathbf{u}) dt \rightarrow \mathcal{J} = \int_0^{\infty} (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt \quad (25)$$

$Q$  and  $R$  are the weighting matrices for the state and control signals, respectively, and are defined by the designer such that  $Q \geq 0$  and  $R > 0$ . The higher the weighting value, the lower the state or control action associated with that value. Regarding the calculation of the matrix  $K$ , it is obtained from the Riccati equation [28], where  $S$  is the Riccati solution to the equation:

$$A^T S + SA - SBR^{-1}B^T S + Q = 0 \rightarrow K = R^{-1}B^T S. \quad (26)$$

Since the longitudinal and lateral direction motions are coupled during flight, it is necessary to separate them in order to develop an appropriate LQR controller. Hence, four different subcontrollers are created so as to fulfill the requirements, so that four feedback gains can be obtained:  $K_{1z}$ , related to the altitude;  $K_{1V}$ , related to the speed;  $K_{21}$  and  $K_{22}$ , related to the tracking of the roll and yaw angles. The reference following  $\phi$  and  $\psi$  is complex; thus, a unique feedback gain does not provide an adequate response, resulting in a requirement to use two feedback gains in a cascade configuration.

In this way,  $K_{1z}$  and  $K_{1V}$  assign the required elevator deflection and throttle lever action to follow the reference in altitude and velocity from the guidance algorithm, respectively. In relation to the roll and yaw angles,  $K_{21}$  creates a reference in angular velocities  $p$  and  $r$  on the basis of the error in these angles with respect to the reference; however, this error is not integrated because a very aggressive and fast action is demanded. Afterward, the feedback gain  $K_{22}$  is applied to this error. The state and control variables and cost functions corresponding to each feedback gain matrix are given by the following expression:

$$\bar{\mathbf{x}}_{1z} = (\Delta z \quad \Delta u \quad \Delta w \quad \Delta \theta \quad \Delta q \quad \epsilon_z)^T, \mathbf{u}_{1z} = (\Delta \delta_E) \Rightarrow \mathcal{J}_1 = \int_0^{\infty} (\bar{\mathbf{x}}_{1z}^T Q_{1z} \bar{\mathbf{x}}_{1z} + \mathbf{u}_{1z}^T R_{1z} \mathbf{u}_{1z}) dt \quad (27)$$

$$\bar{\mathbf{x}}_{1V} = (\Delta V \quad \epsilon_V)^T, \mathbf{u}_{1V} = (\Delta \delta_E) \Rightarrow \mathcal{J}_2 = \int_0^{\infty} (\bar{\mathbf{x}}_{1V}^T Q_{1V} \bar{\mathbf{x}}_{1V} + \mathbf{u}_{1V}^T R_{1V} \mathbf{u}_{1V}) dt \quad (28)$$

$$\bar{\mathbf{x}}_{21} = (\Delta \phi \quad \Delta \psi \quad \epsilon_\phi \quad \epsilon_\psi)^T, \mathbf{u}_{21} = (\Delta p \quad \Delta r)^T \Rightarrow \mathcal{J}_3 = \int_0^{\infty} (\bar{\mathbf{x}}_{21}^T Q_{21} \bar{\mathbf{x}}_{21} + \mathbf{u}_{21}^T R_{21} \mathbf{u}_{21}) dt \quad (29)$$

$$\bar{\mathbf{x}}_{22} = (\Delta p \quad \Delta r \quad \epsilon_p \quad \epsilon_r)^T, \mathbf{u}_{22} = (\Delta \delta_A \quad \Delta \delta_R)^T \Rightarrow \mathcal{J}_4 = \int_0^{\infty} (\bar{\mathbf{x}}_{22}^T Q_{22} \bar{\mathbf{x}}_{22} + \mathbf{u}_{22}^T R_{22} \mathbf{u}_{22}) dt. \quad (30)$$

In order to build these matrices, amplified subsystem matrices must be created. These matrices include the errors ( $\epsilon$ ) in the variables to be reduced. Hence, the state variable vectors comprise an amplified vector, which also contains the variables of error. In addition, matrix  $Q$  presents in its diagonal the weighting factor for each variable of the new amplified state vector. As can be seen in Equation (A5), in matrix  $Q$ , the elements corresponding to the error variables are much more important for prioritizing the tracking error and for diminishing it as much as possible. Moreover, the elements from matrix  $R$  are considerable in size, since only necessary control actions should be used instead of excessive ones. However, these values are limited because they could be unreasonable as a result of poor control actions; therefore, there is an equilibrium between the  $Q$  and  $R$  matrices that provides balanced control responses.

## 5.2. Model Predictive Control

MPC is a strategy that uses an explicit model of the process where, at each sampling time and with the current information of the process variables, predictions of future process behavior along a horizon are managed. Such predictions are incorporated into a cost index



to solve an open-loop optimal control problem (OCP) that is subject to constraints, which results in the sequence of future optimal controls.

Consider the discrete linear system (31), where variables  $\mathbf{x}_{k+i}$  and  $\mathbf{u}_{k+i}$  represent the state vectors and system inputs, respectively, at the instant  $k+i$ ;  $A$  is the state matrix and  $B$  is the system input matrix

$$\mathbf{x}_{k+i+1} = A\mathbf{x}_{k+i} + B\mathbf{u}_{k+i}. \quad (31)$$

For every instant  $k$ , based on (31) and with the availability of the current state of the plant  $\hat{\mathbf{x}}_k$ , predictions are made for the states  $\mathbf{x}_{k+i+1|k}$  and inputs  $\mathbf{u}_{k+i|k}$  along a prediction horizon  $N$ ,  $\forall i \in \{0, 1, \dots, N-1\}$ . These predictions are included in the quadratic cost index (32), which is subsequently minimized at the OCP (33):

$$\mathcal{J}_N(\hat{\mathbf{x}}_k, \mathbf{u}_k) = \sum_{i=0}^{N-1} \left( \mathbf{x}_{k+i|k}^\top Q \mathbf{x}_{k+i|k} + \mathbf{u}_{k+i|k}^\top R \mathbf{u}_{k+i|k} \right) + \mathbf{x}_{k+N|k}^\top P_N \mathbf{x}_{k+N|k} \quad (32)$$

$$\min_{\mathbf{u}_{k|k}, \mathbf{u}_{k+1|k}, \dots, \mathbf{u}_{k+N-1|k}} \mathcal{J}_N(\hat{\mathbf{x}}_k, \mathbf{u}_k) \quad (33)$$

s.t.

$$\mathbf{x}_{k+i+1|k} = A_{cl}\mathbf{x}_{k+i|k} + B\mathbf{u}_{k+i|k} \quad (34)$$

$$H\mathbf{x}_{k+i+1|k} \leq h \quad (35)$$

$$D\mathbf{u}_{k+i|k} \leq d \quad (36)$$

$$\mathbf{x}_{k|k} = \hat{\mathbf{x}}_k \quad (37)$$

$$\forall i \in \{0, 1, \dots, N-1\}, \quad (38)$$

where the current state  $\hat{\mathbf{x}}_k$  is known; thus,  $\mathbf{x}_{k|k} = \hat{\mathbf{x}}_k$ ; matrices  $Q$ ,  $R$ , and  $P_N$  are weighting matrices that penalize the first  $N$  predicted, predicted inputs, and terminal state, respectively. The matrices  $Q$  and  $R$  (equivalent to matrices  $Q$  and  $R$  in LQR controllers) are defined by the designer such that  $Q \geq 0$  and  $R > 0$ , and  $P_N$  is obtained from a quadratic stability analysis [65–67]. Notice that the subscript  $k+i|k$  indicates the predicted value of the variable for the instant  $k+i$  based on the information available at time  $k$ .

In each control period, the problem stated in (33) is solved, obtaining the vector of optimal decision variables  $\mathbf{u}_k^* = \{\mathbf{u}_{k|k}^*, \mathbf{u}_{k+1|k}^*, \dots, \mathbf{u}_{k+N-1|k}^*\}$ , which drives the system states to a desired operating point [68] or as a regulator towards the origin. Next, applying the receding horizon strategy, only the first element  $\mathbf{u}_{k|k}^*$  from the control sequence  $\mathbf{u}_k^*$  is used, so that OCP (33) is performed again at time  $k+1$ .

The cost function (32) obeys the dual-mode prediction paradigm [66,69,70], which ensures stability for an appropriate (or long enough) horizon  $N$ , incorporating a terminal cost  $\mathbf{x}_{k+N|k}^\top P_N \mathbf{x}_{k+N|k}$  that penalizes the terminal state  $\mathbf{x}_{k+N|k}$ . Another condition that is also used to ensure closed-loop stability is to complement the dual mode by adding the terminal constraint  $\mathbf{x}_{k+N|k}$  [69,71] to the OCP (33). The purpose of this is to force the terminal state  $\mathbf{x}_{k+N|k}$  and the following states to remain within a safe zone or terminal set [65,72,73].

The OCP (33) is stated as subject to constraints (34)–(38). Notice that constraint (34) indicates that the predicted state  $\mathbf{x}_{k+i+1|k}$  is computed recursively using the state and inputs from the previous state  $k+i|k$ , where  $\mathbf{x}_{k|k} = \hat{\mathbf{x}}_k$ . Linear inequalities (35) and (36) are constraints for the predicted states and input trajectories, respectively, where matrices  $H$  and  $D$  and vectors  $h$  and  $d$  represent the constraint limits.

Given the quadratic and convex nature of (33), the linear model (31), and the types of constraints (35) and (36), a finite-horizon OCP stated as in (33) can be solved at each control period. This OCP is a quadratic programming problem (QP) with a global optimum whose solution results in the optimal control vector.

In addition to the theoretical formulation, Figure 9 intuitively shows how the MPC works.

The control horizon must always be lower than the prediction horizon. Furthermore, an adequate combination of sample time and control and prediction horizons should be

made, since a low control horizon or a high sample time can lead to poor control actions and vice versa, while low prediction horizon values could demand excessive or even impossible control actions.

The prediction and control horizon parameters are set to 100 and 10, respectively. Therefore, at each iteration, the controller calculates the performance of the system for a total of  $t_p = 100 \cdot T_s = 2.1$  s by means of applying control actions until time  $t_c = 10 \cdot T_s = 0.21$  s, selecting the best options among all of the calculations.

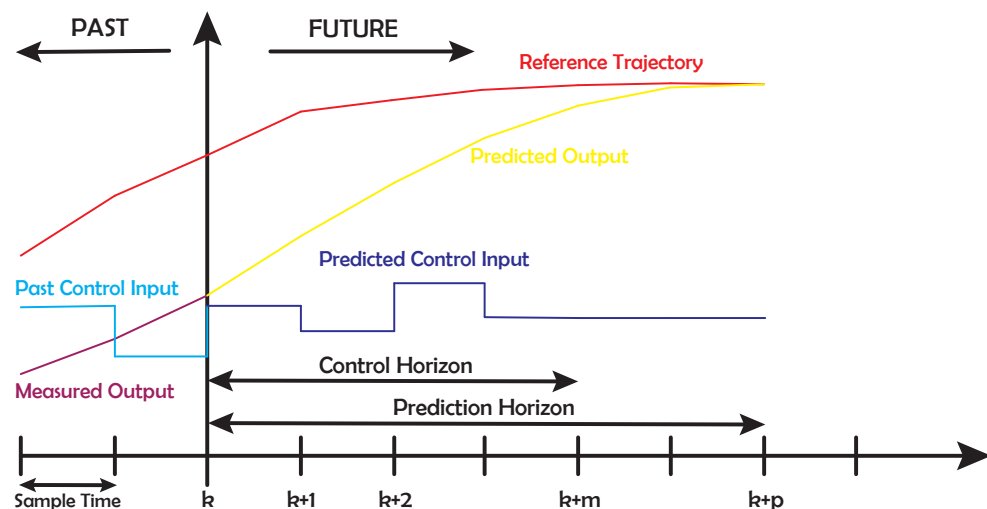


Figure 9. Model predictive control (MPC) future prediction scheme.

Additionally, the physical constraints shown in [40] are introduced into the MPC following the OCP structure. This is because it is not necessary to include a signal filter after the control block, as is done in the LQR controller.

The controller responses are determined by the weighting matrices  $Q$  and  $R$  applied to the state variables and control variables. Tracking is only performed for the altitude, velocity, and roll and yaw angles.

As shown in Equation (A6), some variables are specifically high in the weighting matrices  $Q$  and  $R$ ; the pitch angle, angular velocity  $q$ , and elevator deflection are prioritized to avoid excessive motion in the vertical plane and altitude errors.

## 6. Results

### 6.1. Complete Response

The flight simulations utilizing both controllers, i.e., LQR and MPC, were performed under variable mass and wind conditions, as explained in Section 2. The wind velocities in each axis obtained during the simulations are shown in Figure A1. The compressibility effects are not given, since the divergence Mach number was not exceeded.

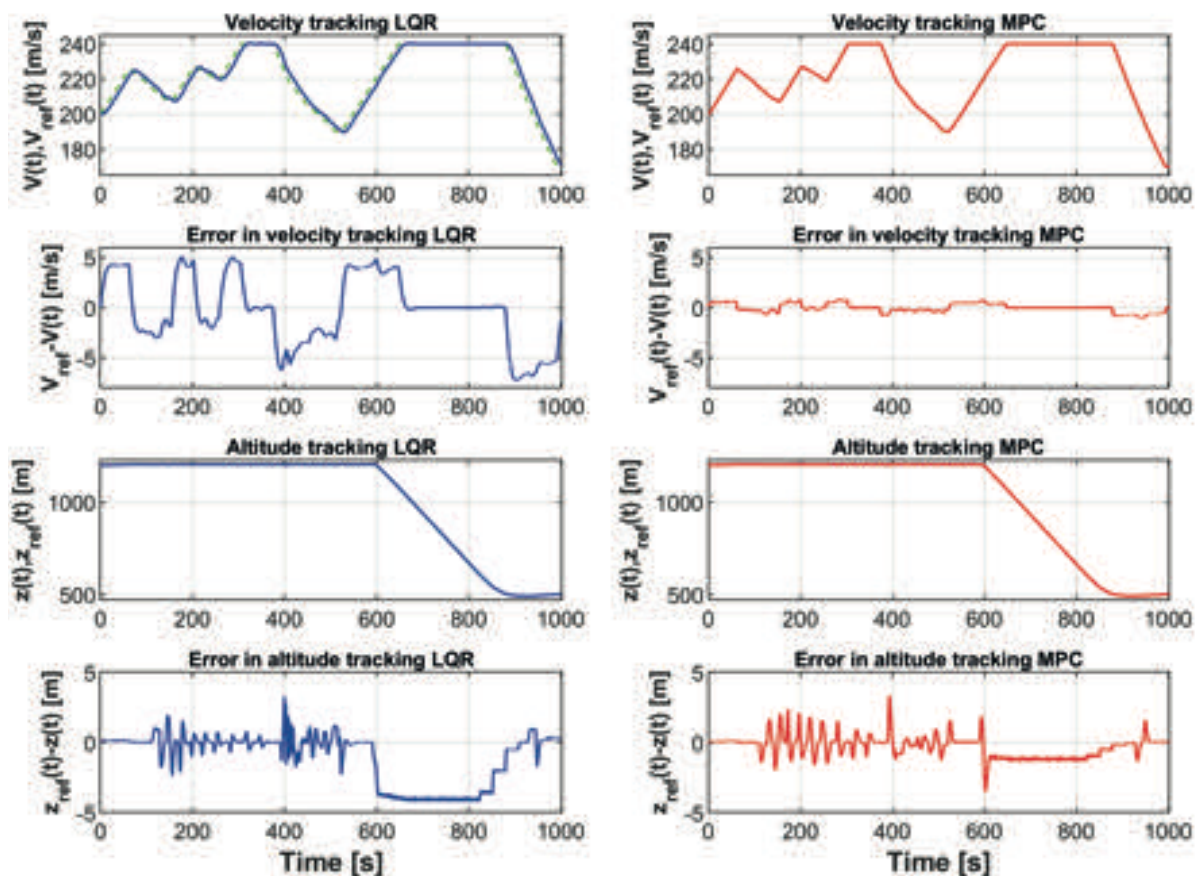
The code developed to perform all the simulations is available on Matlab Central [74], including the path-planning algorithms, control parameter definitions, and Simulink files, in which the controllers and guidance law that were employed in simulations are implemented.

In Figure 10, the velocity and altitude magnitudes are shown, together with their respective reference values specified by the dynamic trajectory. The MPC case clearly provided a more satisfactory response in velocity tracking, while the reference altitude was followed similarly in both cases. Nevertheless, the LQR controller showed several problems during curves and descents, since the velocities and altitudes presented considerable errors, while the MPC controller achieved a more satisfactory performance.

When great differences with respect to the equilibrium point were given in the flight path and roll angle (in the state-space model,  $\gamma$  and  $\phi$  were considered null due to the trimmed point conditions), such as in intense descents or curves, the longitudinal and

lateral direction motions were coupled to a considerable extent, which provoked acute errors in the reference tracking, since the LQR and linear time-invariant MPC controllers worked continuously with the same plant model. In this case, both of the controllers' performance during curves showed appropriate robustness, although the MPC controller more adequately tracked the reference velocity.

Generally, an MPC controller creates more adequate control actions than an LQR controller due to its future prediction capabilities; however, both controllers really depend on the selection of the weighting parameters. In the case of the MPC controller, their suitable selection, which restricts the pitching motion and assigns a high priority to altitude and velocity tracking, provided satisfactory results during the whole simulation. Regarding the LQR case, the parameters of the  $Q$  and  $R$  matrices and the structure of this controller were selected appropriately. The special configuration on the part of the LQR controller corresponding to the roll and yaw angles led to stronger capabilities of tracking the trajectory.



**Figure 10.** Reference tracking of velocity and altitude: linear quadratic regulator (LQR) (blue), MPC (red), and the reference for each case (discontinuous green).

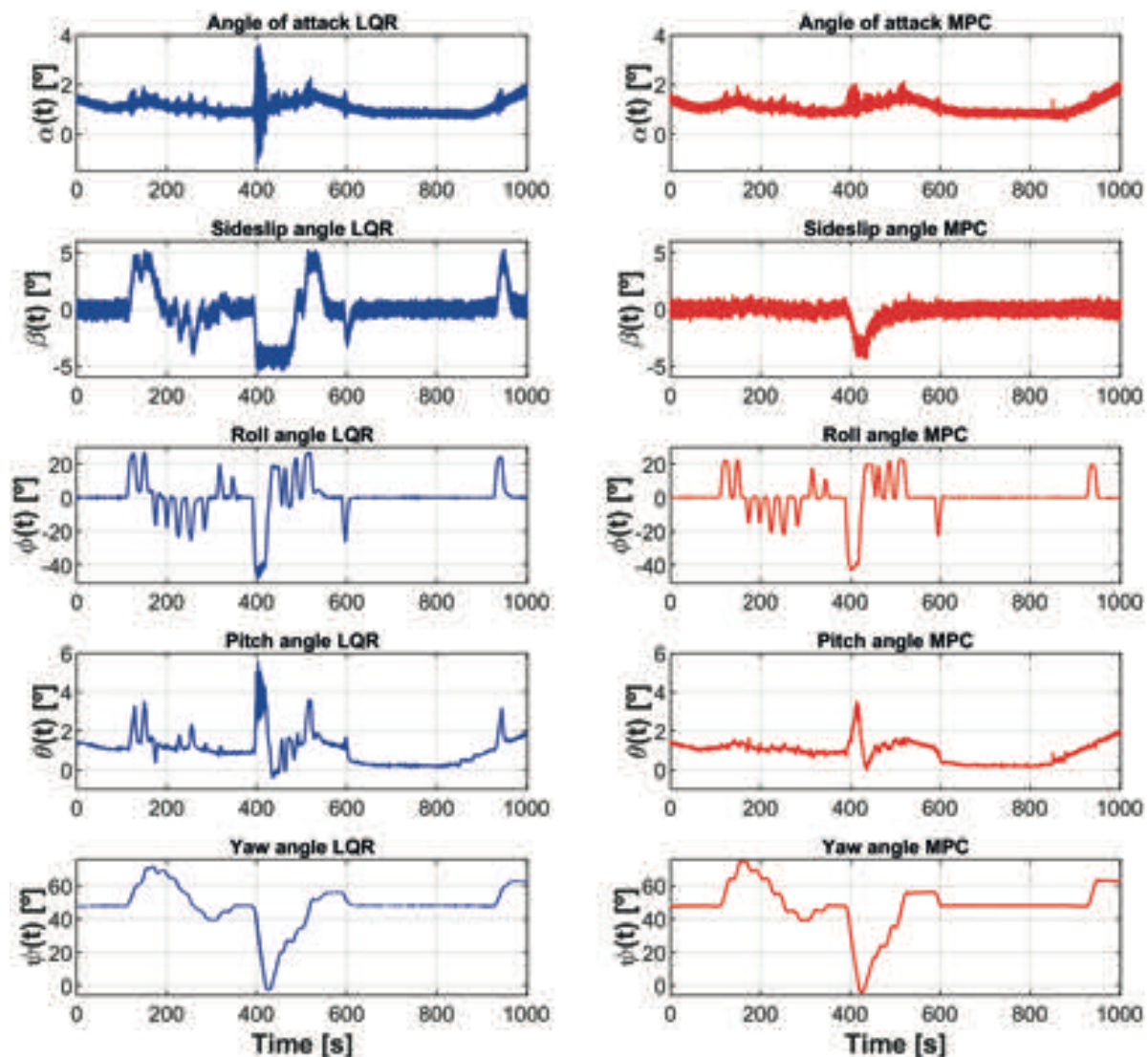
In Figure 11, several of the principal flight parameters are shown: the angle of attack, sideslip angle, and Euler angles. As can be appreciated, the fluctuations are fairly intense in all variables due to the wind, although they are especially relevant in the sideslip angle.

The angle of attack presented realistic values in almost all of the trajectory, but in some points, it entered into the negative area due to aggressive control actions, which is correlated to the error in altitude tracking shown in Figure 10. These negative values enlarged the probability of stalling in the real behavior of the UAV, so that the LQR controller could produce several failures in this sense, such as, for example, at a time of  $t \approx 400$  s.

With regard to the sideslip angle, in no cases was it null because the wind velocity restricts the yaw and course angle from coinciding in straight lines. Also, this parameter

showed very well the regions where the turns occurred. Concerning to the notable curves, if they are made in the clockwise direction, the sideslip should be mainly positive; on the contrary, counterclockwise curves provoke negative sideslip angles. Therefore, the LQR results present a more clear development of the aircraft rotation for proceeding with curves properly, while the MPC results represent a more erratic behavior.

Considering to the pitch angle, the motion was more aggressive in the LQR case because the elevator control action was not as anticipatory as that of the MPC controller, which demonstrated excessive oscillations in the angle of attack and altitude tracking errors. Last but not least, the roll and yaw angles were very similar in both cases; however, they presented lower fluctuations and overshoots in the MPC case. Therefore, the guidance law acted slightly more ineffectively for the LQR controller, or the controller was not robust enough in the lateral direction motion.



**Figure 11.** Principal flight parameters by LQR (blue) and MPC (red): the angle of attack ( $\alpha$ ), sideslip angle ( $\beta$ ), and Euler angles ( $\phi$ ,  $\theta$ ,  $\psi$ ).

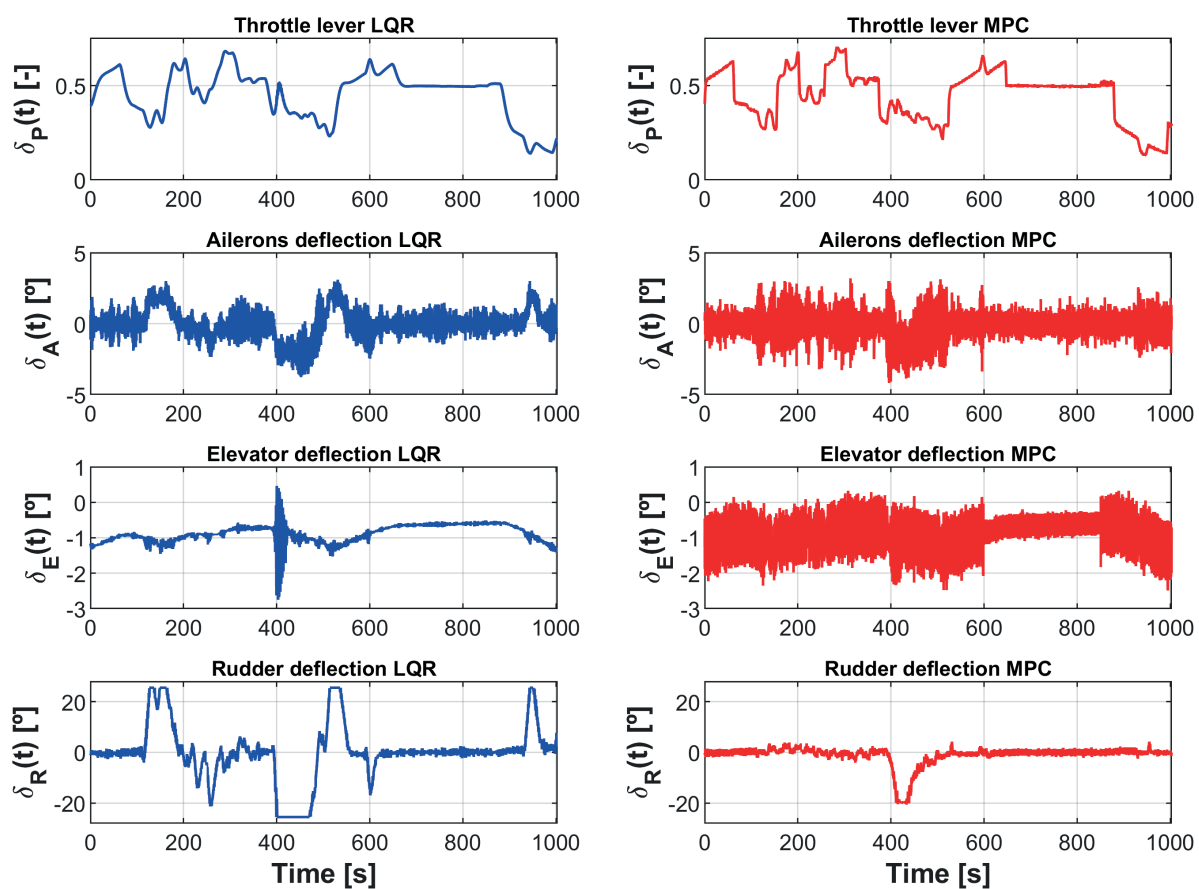
The control action signals are given in Figure 12. More intense fluctuations in the control actions were obtained in the MPC results, which confirms the previous results, in which the flight parameters oscillated less. The amplitude of the oscillations in the control signals of the rudder was primarily higher in curves in the LQR case, demonstrating a more aggressive control for facing perturbations and adequately tracking the reference path. In



relation to the ailerons and elevator deflections, regarding the former, their fluctuations were comparable in both cases; meanwhile, regarding the latter, their oscillations were chiefly more intense in the MPC case, although their average value was very close to that of the LQR controller, owing to the improvement of the stability of the aircraft by the greater robustness of this controller.

The error in altitude can be explained by the non-coordinated control actions of the elevator because the controllers utilized it without taking into account that it also interferes in the lateral direction motion because of the linearization process with respect to the trimmed point.

In addition, the mean value of the oscillations in the rudder along the straight path was not zero. This was due to the fact that the course angle was not exactly the one specified by the guidance algorithm and the sideslip angle fluctuated with respect to an angle higher than zero (see Figure 11). This problem was less significant in the MPC case, which could be due to the well-coordinated control actions between the rudder and ailerons.



**Figure 12.** Control actions by the LQR (blue) and MPC (red) controllers: the throttle lever ( $\delta_P$ ) and elevator ( $\delta_E$ ), ailerons ( $\delta_A$ ), and rudder ( $\delta_R$ ) deflections.

Finally, in Figure 13, the trajectories of both cases are shown. The errors in altitude were similar in the LQR and MPC cases, as can be seen in the figure. In relation to the tracking of the path in the XY plane, both controllers provided satisfactory results, since the deviation was minimum; nevertheless, if more intense curves were given, the LQR would accomplish a slightly more appropriate planar tracking due to its adequate coordination of the control signals, as shown by the sideslip angle results in Figure 11. However, the clear advantage of the MPC was in the velocity results, as can be observed in Figure 10.



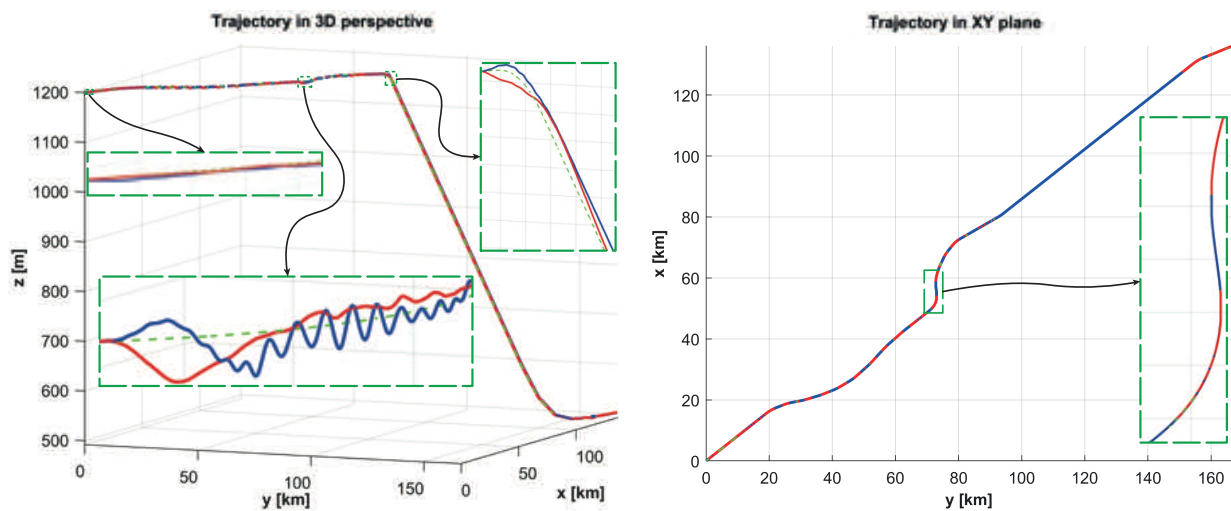


Figure 13. Final trajectory from two perspectives; LQR (blue), MPC (red), and reference path (discontinuous green).

### 6.2. Analysis of the Impact, Time, and Fuel Mass Consumption

By means of a dropping point estimation algorithm [45], which calculates the best location of the trajectory to release the bombs and their deviations in the impact, based on a mathematical model of M117 bombs [43,75] (detailed in [45]), the deviations in the dropping point from the reference release coordinates and the error in the impact—defined by the expressions (39) and (39)—were calculated in the LQR and MPC cases. This mathematical model assumes no gyration of the bombs around their x-axis; hence, only the yaw and pitch angles can vary. Furthermore, the mean wind velocity and direction were considered in the calculation of the reference case, while the normalized data were utilized in the real results.

$$Deviation_{Dropping} = \sqrt{(x_{Drop,ref} - x_{Drop})^2 + (y_{Drop,ref} - y_{Drop})^2} \quad (39)$$

$$Deviation_{Impact} = \sqrt{(x_{Impact} - q_{f,x})^2 + (y_{Impact} - q_{f,y})^2} \quad (40)$$

Figure 14 represents an example of the application of the dropping point estimation algorithm. The bomb trajectories were calculated from each waypoint. Afterward, the most adequate dropping coordinates were obtained by means of diminishing the deviations in the bombs’ impacts, which were determined through the collision on the ground. The terrain was modeled through a bilinear interpolation [76].

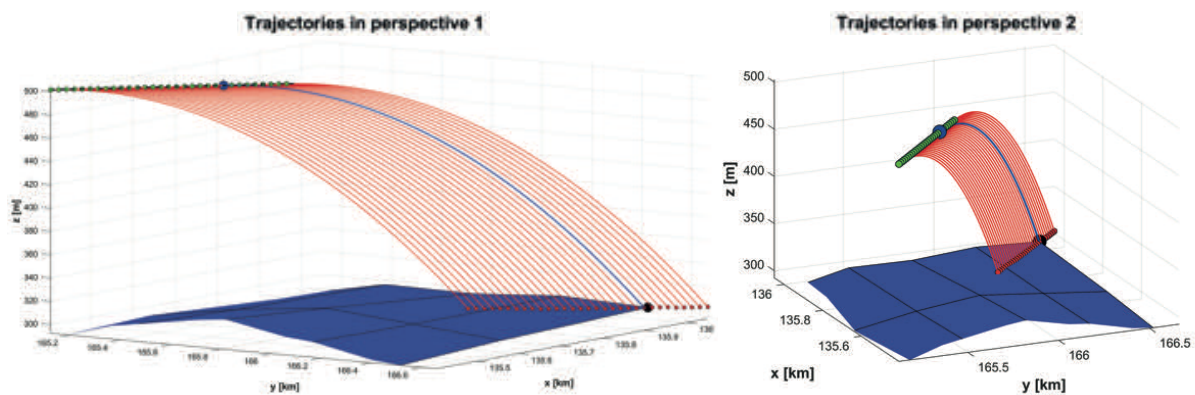


Figure 14. Dropping point estimation applied at the reference case: possible options (red lines) from suboptimal dropping points (green dots) and most appropriate option (blue line) from optimal release point (blue dot).

As can be seen in Table 1, the MPC controller offered more adequate results, since the deviations in the dropping point and error in the impact were fairly low, while the LQR’s dropping point was a considerable distance from the reference one, which means that the

impact could be particularly inaccurate in terms of the real behavior. This can be explained by the appropriate velocity tracking by the MPC controller, since the error in speed during the drop was greater in the LQR case.

**Table 1.** Deviations in the release (from reference case) and in the impact of the bombs.

Case	Deviations	
	Deviation in the Drop (m)	Deviation in Impact (m)
Reference	-	7.7
LQR	14.3	4.0
MPC	3.0	1.6

However, these deviations could materially vary due to the fact that the bomb-dropping results are fairly contingent upon the final part of the dynamic trajectory. However, the velocity tracking of the MPC controller is still bound to guarantee a more appropriate response.

In addition, the flight time and fuel consumption were compared in order to evaluate the performance during the mission, since the relevant differences in these magnitudes could determine a greater efficiency of one controller over the other in the net results and because these magnitudes also provide a bigger picture of flight autonomy.

The flight time was slightly smaller in the LQR case, while the fuel consumption was lower in the MPC case, as can be seen in Table 2, although the differences were not significant. Hence, both controllers would be equally useful for missions.

**Table 2.** Simulation time and fuel consumption.

Case	Time and Fuel Mass	
	Simulation Time (s)	Fuel Consumption (kg)
LQR	1001.09	266.77
MPC	1002.67	264.08

## 7. Conclusions and Future Works

A realistic simulation environment was implemented for the F-86 Sabre acting as a UAV based on a group of known algorithms to analyze which multivariable control method provides a more satisfactory performance. This system was able to fulfill all of the mission's requirements according to Section 2.

The dynamic trajectory-planning algorithms, which are modified versions of those in [14,45,62], demonstrated a proper behavior in the creation of waypoints for trajectories that are perfectly trackable for the aircraft and allow one to avert 3D obstacles. Furthermore, the 3D smooth path-planning code facilitated the reference-following task of the controllers. The reasonable velocity profile and the unusual approach of the neighborhood finding turned out to be appropriate for the path due to the fact that both controllers were able to perform adequate reference tracking. However, an improvement in these elements, as well as in the rewarding function and the selection of the center of the rectangloids, will be key to making further progress in dynamic trajectory building in future designs. For instance, the rewarding function could consider new flight parameters in its calculations and the center selections could take their proximity to obstacles into account to make the route more secure.

The LQR and MPC controllers provided acceptable responses, since the aircraft did not enter prohibited areas in either case, and the aircraft's limitations were respected; nevertheless, the trajectory obtained by the MPC method was more satisfactory in terms of airworthiness.

Regarding the guidance law, a nonlinear one that improves the proportional guidance capabilities is a particularly appropriate option. All of the references in altitude, velocity,

and roll and yaw angles were adequate; hence, the path tracking in the horizontal plane presented low deviations. In future designs, predictive guidance laws created from motion equations could be implemented to better enhance the flying system.

Finally, the bombing was accurate in the MPC case, while the LQR results presented noticeable deviations, even though the differences were not great, since they are dependent on many factors. The excellent reference velocity following of the MPC controller led to a more satisfactory performance in terms of the bombs' impact, but also in the fuel consumption. The predictive capabilities of the MPC provide a more optimal control than in the LQR case, as the control signals' configuration is managed more efficiently. Consequently, the MPC controller could be considered a more appropriate option, since it provided acceptable results in terms of obstacle avoidance and reference tracking and achieved only a slight error during the blitz operation.

Two main aspects are open for future works: on one hand, the specific procedures for landing and takeoff maneuvers, and on the other hand, reliable hardware for implementing the proposed algorithms.

Regarding the takeoff, factors related to the runway, such as the friction coefficient, the altitude of its location, or the local temperature, can have significant effects, while the ground effect (GE) [77] substantially modifies the lift and drag. However, a simple physical analysis that takes these aspects into account [43] can be an initial approach.

In relation to landings, the procedure is more awkward. Not only do the runway characteristics and the GE disrupt the usual flight motion, but crosswind and attitude errors are also particularly noteworthy. Therefore, new elements that serve as aids—namely, an instrument landing system (ILS) [78] or image processing techniques [79]—can be implemented to facilitate the operation in this flight phase. Both methodologies provide an important level of autonomy, although it could be said that the ILS is more reliable due to the fact that errors in the image-taking process due to the weather could be disastrous at typical landing velocities. Nevertheless, image processing would be ideal for other aircraft, such as helicopters or quadcopters, that are able to achieve a fixed-point flight.

Finally, real implementation is subject to reliable and high-powered hardware. Recent research, such as that of [80] or [81], suggests that the best alternatives today are systems based on field-programmable gate arrays (FPGAs), which adapt to the demanding requirements of the aerospace sector without losing flexibility and computing power.

**Author Contributions:** Conceptualization, A.O., S.G.-N., and R.S.; methodology, A.O.; software, A.O.; formal analysis, A.O.; investigation, A.O. and S.G.-N.; writing—original draft preparation, A.O., S.G.-N., and R.S.; writing—review and editing, A.O., S.G.-N., and R.S.; visualization, A.O.; supervision, S.G.-N. and R.S.; project administration, S.G.-N.; funding acquisition, S.G.-N. and R.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially funded by project RTI2018-096904-B-I00 from the Spanish Ministry of Economy and by project AICO/2019/055 from Generalitat Valenciana.

**Acknowledgments:** The authors would like to thank the editors and the reviewers for their valuable time and constructive comments.

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

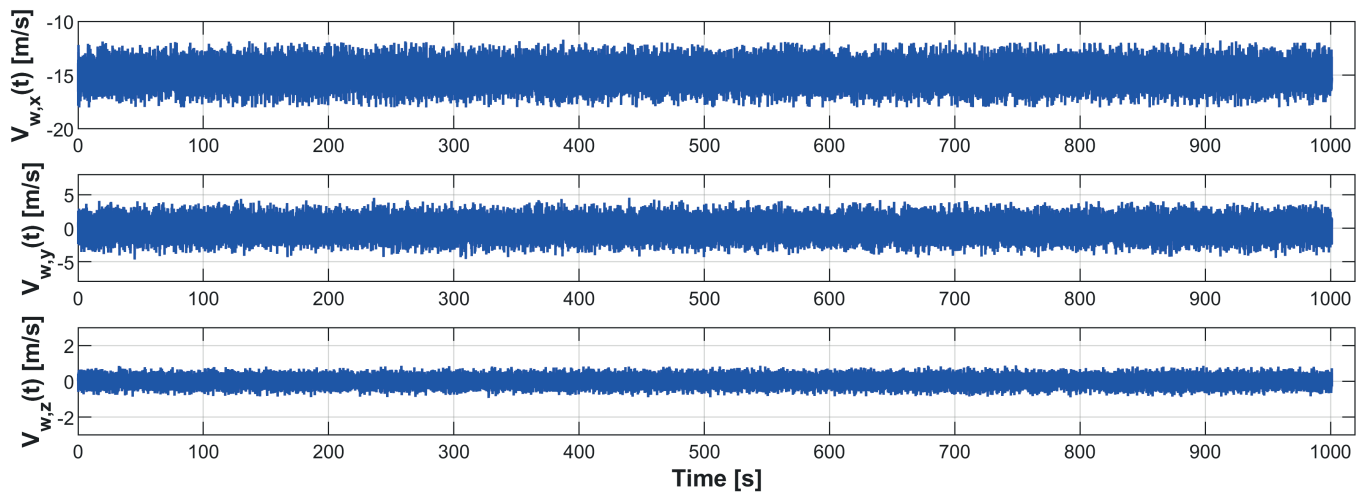
### Appendix A. Wind Parameters and Mathematical Model

**Table A1.** The wind parameters employed to build the wind model:  $\mu$  (mean),  $\sigma$  (standard deviation), and  $Y_{max}$  (maximum) and  $Y_{min}$  (minimum) values.

Wind Parameters			
Parameter	Modulus $V_{wind}$ (m/s)	Direction Angle $\kappa$ (°)	Elevation Angle $\omega$ (°)
$\mu$	15	180	0
$\sigma$	1	5	1
$Y_{max}$	18	195	3
$Y_{min}$	12	165	-3

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \Rightarrow \text{Normalized Variable : } x, \quad g(x, \mu, \sigma) = \begin{cases} Y_{min} & x < \mu - 3\sigma \\ x & \mu - 3\sigma \leq x \leq \mu + 3\sigma \\ Y_{max} & x > \mu + 3\sigma \end{cases} \quad (A1)$$

$$\begin{cases} V_{wind}(x) = g(x, \mu_V, \sigma_V) \\ \kappa(x) = g(x, \mu_\kappa, \sigma_\kappa) \\ \omega(x) = g(x, \mu_\omega, \sigma_\omega) \end{cases} \quad \begin{cases} V_{w,x}(t) = V_{wind} \cos(\kappa) \cos(\omega) \\ V_{w,y}(t) = V_{wind} \sin(\kappa) \cos(\omega) \\ V_{w,z}(t) = V_{wind} \sin(\omega) \end{cases} \quad (A2)$$



**Figure A1.** Wind velocity in each Cartesian axis.

### Appendix B. Linearized Model of the System

$$A \cdot \Delta \underline{r} = \begin{pmatrix} 0 & 2.21 \times 10^{-2} & 0 & -1.00 & 0 & 215 & 0 & 0 & 0 & 0 & 0 \\ -4.60 \times 10^{-5} & -1.10 \times 10^{-2} & 0 & 5.41 \times 10^{-2} & 0 & -9.81 & 0 & 0 & -4.73 & 0 & 0 \\ 0 & 0 & -1.87 \times 10^{-1} & 0 & 9.81 & 0 & 0 & 4.47 & 0 & -214.03 & 0 \\ 9.18 \times 10^{-4} & -5.81 \times 10^{-2} & 0 & -2.21 & 0 & -2.16 \times 10^{-1} & 0 & 0 & 212.89 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.00 & 0 & 2.21 \times 10^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.00 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & -1.74 \times 10^{-1} & 0 & 0 & 0 & 0 & -2.34 & 0 & 7.59 \times 10^{-1} & 0 \\ -3.19 \times 10^{-6} & 6.00 \times 10^{-3} & 0 & -2.25 \times 10^{-1} & 0 & -1.09 \times 10^{-19} & 0 & 0 & -3.25 & 0 & 0 \\ 0 & 0 & 3.30 \times 10^{-2} & 0 & 0 & 0 & 0 & -6.20 \times 10^{-3} & 0 & -2.08 \times 10^{-1} & 0 \\ -2.57 \times 10^{-5} & 1.12 \times 10^{-2} & 0 & 5.30 \times 10^{-3} & 0 & -9.81 & 0 & 0 & -3.72 \times 10^{-2} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta z \\ \Delta u \\ \Delta v \\ \Delta w \\ \Delta \phi \\ \Delta \theta \\ \Delta \psi \\ \Delta p \\ \Delta q \\ \Delta r \\ \Delta V \end{pmatrix} \quad (A3)$$

$$\bar{B} \cdot \bar{\Delta u} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3.14 & 1.40 \times 10^{-1} & 0 & 0 & 6.20 \times 10^{-3} & 7.60 \times 10^{-3} & -5.43 \times 10^{-2} & -92.48 \\ 0 & 0 & 0 & 4.74 & -1.45 \times 10^{-1} & 1.18 \times 10^{-1} & 0 & 0 \\ 2.17 \times 10^{-4} & -35.53 & 0 & 0 & 6.76 \times 10^{-2} & 8.27 \times 10^{-2} & 2.21 & -2.04 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 51.67 & 2.10 & -1.35 \times 10^{-1} & 1.10 \times 10^{-1} & 0 & 0 \\ 3.40 \times 10^{-4} & -56.33 & 0 & 0 & -6.72 \times 10^{-4} & -8.23 \times 10^{-4} & 2.23 \times 10^{-1} & -2.92 \times 10^{-18} \\ 0 & 0 & 1.62 \times 10^{-2} & -1.17 & 2.56 \times 10^{-2} & -2.09 \times 10^{-2} & 0 & 0 \\ 3.14 & -6.43 \times 10^{-1} & 0 & 0 & 7.70 \times 10^{-3} & 9.40 \times 10^{-3} & -5.50 \times 10^{-3} & -92.50 \end{pmatrix} \begin{pmatrix} \Delta \delta_P \\ \Delta \delta_E \\ \Delta \delta_A \\ \Delta \delta_R \\ V_{x_w} \\ V_{y_w} \\ V_{z_w} \\ CD_{wave} \end{pmatrix} \quad (A4)$$

### Appendix C. LQR Parameters

$$\begin{cases} Q_{1z} = \text{diag}(1 \times 10^{-6}, 4 \times 10^{-4}, 100, 14.59, 131.31, 6 \times 10^{10}), Q_{1V} = \text{diag}(4 \times 10^{-4}, 1 \times 10^6) \\ Q_{21} = \text{diag}(4.05 \times 10^{-1}, 4.05 \times 10^{-1}, 1 \times 10^{13}, 1 \times 10^{13}), Q_{22} = \text{diag}(3.65, 8.21, 1 \times 10^9, 1 \times 10^{10}) \\ R_{1z} = 1.46 \times 10^{15}, R_{1V} = 2.00 \times 10^{10}, R_{21} = \text{diag}(2.92 \times 10^{12}, 1.34 \times 10^{13}), R_{22} = \text{diag}(6.57 \times 10^8, 2.36 \times 10^8) \\ K_{1z} = (-9.70 \times 10^{-3}, -4.18 \times 10^{-4}, 6.20 \times 10^{-3}, -1.57, -5.46 \times 10^{-2}, 6.40 \times 10^{-3}), K_{1V} = (6.68 \times 10^{-2}, -7.10 \times 10^{-3}) \\ K_{21} = \begin{pmatrix} -1.85 & 1.29 \times 10^{-2} \\ -6.00 \times 10^{-3} & -8.64 \times 10^{-1} \end{pmatrix}, K_{22} = \begin{pmatrix} 1.77 \times 10^{-1} & 1.22 \times 10^{-1} & -1.23 & 1.89 \times 10^{-1} \\ 1.33 \times 10^{-2} & -3.15 & -9.98 \times 10^{-2} & 6.50 \end{pmatrix} \end{cases} \quad (A5)$$

### Appendix D. MPC Parameters

$$\begin{cases} Q = \text{diag}(2.50 \times 10^{-1}, 0, 0, 0, 1, 0, 5 \times 10^{-1}, 0, 30, 0, 2.50 \times 10^{-1}) \\ R = \text{diag}(0, 2, 1 \times 10^{-2}, 1.25 \times 10^{-1}) \end{cases} \quad (A6)$$

## References

- Valavanis, K.P.; Vachtsevanos, G.J. Future of Unmanned Aviation. In *Handbook of Unmanned Aerial Vehicles*; Springer: Dordrecht, The Netherlands, 2015; pp. 2993–3009, doi:10.1007/978-90-481-9707-1\_95.
- Microdrones. Unmanned Applied Solutions. 2020. Available online: <https://www.microdrones.com/en/industry-experts/> (accessed on 14 November 2020).
- Roth, M. AI in Military Drones and UAVs—Current Applications. 2019. Available online: <https://emerj.com/ai-sector-overviews/ai-drones-and-uavs-in-the-military-current-applications/> (accessed on 14 November 2020).
- Jerry, M. Current and future UAV military users and applications. *Air Space Eur.* **1999**, *1*, 51–58, doi:10.1016/S1290-0958(00)88871-1.
- Geo-Matching. Aerial Imagery from UAVs. 2020. Available online: <https://geo-matching.com/content/aerial-imagery-from-uavs> (accessed on 14 November 2020).
- SenseFly. Why Use Drones For Environmental Monitoring? 2020. Available online: <https://www.sensefly.com/industry/drones-environmental-monitoring/> (accessed on 14 November 2020).
- EKU. 5 Ways Drones Are Being Used for Disaster Relief. 2020. Available online: <https://safetymanagement.eku.edu/blog/5-ways-drones-are-being-used-for-disaster-relief/> (accessed on 14 November 2020).
- Boeing. F-86 Sabre Jet—Historical Snapshot. 2020. Available online: <https://www.boeing.com/history/products/f-86-sabre-jet.page> (accessed on 14 November 2020).
- The Editors of Encyclopaedia Britannica. F-86. 2019. Available online: <https://www.britannica.com/technology/F-86> (accessed on 14 November 2020).
- X-47B UCAS Unmanned Combat Air System. 2015. Available online: <https://www.northropgrumman.com/what-we-do/air/x-47b-ucas/> (accessed on 14 November 2020).
- NavalTechnology. X-47B Unmanned Combat Air System (UCAS). 2016. Available online: <https://www.naval-technology.com/projects/x-47b-unmanned-combat-air-system-carrier-ucas/> (accessed on 14 November 2020).
- Luo, C.; McClean, S.I.; Parr, G.; Teacy, L.; De Nardi, R. UAV position estimation and collision avoidance using the extended Kalman filter. *IEEE Trans. Veh. Technol.* **2013**, *62*, 2749–2762.
- LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006; pp. 1–826, doi:10.1017/CBO9780511546877.
- Samaniego, F.; Sanchis, J.; Garcia-Nieto, S.; Simarro, R. Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs. *Electronics* **2019**, *8*, 306, doi:10.3390/electronics8030306.
- Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271, doi:10.1007/BF01386390.



16. Niu, L.; Zhuo, G. An Improved Real 3D a\* Algorithm for Difficult Path Finding Situation. In Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, China, 3–11 July 2008; pp. 927–930.
17. Ferguson, D.; Stentz, A. Field D\*: An Interpolation-Based Path Planner and Replanner. In *Robotics Research: Results of the 12th International Symposium, October 12–15, 2005, San Francisco, CA, USA*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 28, pp. 239–253, doi:10.1007/978-3-540-48113-3\_22.
18. De Filippis, L.; Guglieri, G.; Quagliotti, F. Path Planning Strategies for UAVS in 3D Environments. *J. Intell. Robot. Syst.* **2012**, *65*, 247–264, doi:10.1007/s10846-011-9568-2.
19. Nakajima, K.; Premachandra, C.; Kato, K. 3D environment mapping and self-position estimation by a small flying robot mounted with a movable ultrasonic range sensor. *J. Electr. Syst. Inf. Technol.* **2017**, *4*, 289–298, doi:10.1016/j.jesit.2017.01.007.
20. Premachandra, C.; Otsuka, M.; Gohara, R.; Ninomiya, T.; Kato, K. A study on development of a hybrid aerial/terrestrial robot system for avoiding ground obstacles by flight. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 327–336, doi:10.1109/JAS.2018.7511258.
21. Premachandra, C.; Thanh, D.N.H.; Kimura, T.; Kawanaka, H. A study on hovering control of small aerial robot by sensing existing floor features. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1016–1025, doi:10.1109/JAS.2020.1003240.
22. Marco, V.; Contreras, R.; Sanchez, R.; Rodriguez, G.; Serrano, D.; Kerr, M.; Fernandez, V.; Haya-Ramos, R.; Peñin, L.F.; Ospina, J.A.; et al. The IXV guidance, navigation and control subsystem: Development, verification and performances. *Acta Astronaut.* **2016**, *124*, 53–66, doi:10.1016/j.actaastro.2016.04.010.
23. Premachandra, C.; Ueda, D.; Kato, K. Speed-Up Automatic Quadcopter Position Detection by Sensing Propeller Rotation. *IEEE Sens. J.* **2019**, *19*, 2758–2766, doi:10.1109/JSEN.2018.2888909.
24. Premachandra, C.; Otsuka, M. Development of hybrid aerial/terrestrial robot system and its automation. In Proceedings of the 2017 IEEE International Systems Engineering Symposium (ISSE), Vienna, Austria, 11–13 October 2017; pp. 1–3, doi:10.1109/SysEng.2017.8088328.
25. Pardo, C. Controlador PID. 2020. Available online: <https://www.picuinio.com/es/arduprog/control-pid.html> (accessed on 14 November 2020).
26. Fadali, M.S.; Visioli, A. Chapter 9—State Feedback Control. In *Digital Control Engineering*, 2nd ed.; Fadali, M.S., Visioli, A., Eds.; Academic Press: Boston, MA, USA, 2013; pp. 351–397, doi:10.1016/B978-0-12-394391-0.00009-5.
27. Mathworks. Lqr. 2020. Available online: <https://www.mathworks.com/help/control/ref/lqr.html> (accessed on 14 November 2020).
28. Hajiyev, C.; Soken, H.; Vural, S. Linear Quadratic Regulator Controller Design. In *State Estimation and Control for Low-Cost Unmanned Aerial Vehicles*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 171–200, doi:10.1007/978-3-319-16417-5\_10.
29. Wang, L. *Model Predictive Control System Design and Implementation Using MATLAB*; Springer: London, UK, 2009; pp. 1–332, doi:10.1007/978-1-84882-331-0.
30. Grancharova, A.; Johansen, T. Nonlinear Model Predictive Control. In *Explicit Nonlinear Model Predictive Control*; Grancharova, A., Johansen, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 39–63, doi:10.1007/978-3-642-28780-0\_2.
31. Zhou, K.; Doyle, J.C. *Essentials of Robust Control*; Prentice hall Upper Saddle River: New York, USA, 1998; pp. 1–10.
32. Gavilán, F.R. *Sistemas de Control y Guiado para Vehículos Aéreos No Tripulados, Diseño de Algoritmos y Sistemas Embarcados*. Ph.D. Thesis, University of Sevilla, Sevilla, Andalusia, Spain, 2012.
33. Sujit, P.B.; Saripalli, S.; Sousa, J.B. An evaluation of UAV path following algorithms. In Proceedings of the 2013 European Control Conference (ECC), Zurich, Switzerland, 17–19 July 2013; pp. 3332–3337, doi:10.23919/ECC.2013.6669680.
34. Espinoza-Fraire, T.; Dzul, A.; Cortés-Martínez, F.; Giernacki, W. Real-time implementation and flight tests using linear and nonlinear controllers for a fixed-wing miniature aerial vehicle (MAV). *Int. J. Control Autom. Syst.* **2018**, *16*, 392–396.
35. Bulka, E.; Nahon, M. Autonomous fixed-wing aerobatics: from theory to flight. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6573–6580.
36. Kamel, M.; Stastny, T.; Alexis, K.; Siegwart, R. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In *Robot Operating System (ROS)*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 3–39.
37. Stastny, T.J.; Dash, A.; Siegwart, R. Nonlinear MPC for fixed-wing UAV trajectory tracking: Implementation and flight experiments. In Proceedings of the Aiaa Guidance, Navigation, and Control Conference, Grapevine, TX, USA, 9–13 January 2017; p. 1512.
38. Department of Defense of the United States of America. OP 1280. Declassification. 1955. Available online: <https://fdocuments.in/document/op-1280-aircraft-bombs.html> (accessed on 14 November 2020).
39. United States Air Force. Handbook Maintenance Instructions USAF Series F-86F Aircraft. Declassification. 1954. Available online: <https://www.avialogs.com/aircraft-n/north-american-aviation/item/55493-t-o-1f-86f-2-handbook-maintenance-instructions-f-86f-aircraft> (accessed on 14 November 2020).
40. United States Air Force. Flight Manual USAF Series F-86F Aircraft. Declassification. 1960. Available online: <https://www.esscoaircraft.com/p-9741-north-american-f-86f-1960-flight-manual.aspx> (accessed on 14 November 2020).
41. Cook, M. *Flight Dynamics Principles*, 3rd ed.; Butterworth-Heinemann: Cranfield, UK, 2013; pp. 1–145, doi:10.1016/B978-0-08-098242-7.00003-1.
42. Napolitano, M. *Aircraft Dynamics: From Modeling to Simulation*; John Wiley & Sons, Inc.: Morgantown, WV, USA, 2012; pp. 135–267.
43. Magraner Rullan, J.P.; Carreres Talens, M.; Martí Gómez-Aldaraví, P. *Flight Mechanics*; Universitat Politècnica de València: Valencia, Spain, 2018; pp. 1–190, doi:10.1007/978-981-10-8721-9\_2.
44. Roux, É. *Turbofan and Turbojet Engines: Database Handbook*; Élodie Roux: Blagnac, France, 2007; p. 242.

45. Ortiz, Á. Design and Simulation of the Guidance and Control System for an F-86 Sabre. Bachelor's Thesis, Universitat Politècnica de València, Valencian Community, Spain. 2020. Available online: <http://hdl.handle.net/10251/153251> (accessed on 14 November 2020).
46. Sumair, M.; Aized, T.; Gardezi, S.A.R.; Bhutta, M.M.A.; Rehman, S.M.S.; ur Rehman, S.U. Application of five continuous distributions and evaluation of wind potential at five stations using normal distribution. *Energy Explor. Exploit.* **2020**, 1–3, doi:10.1177/0144598720939373.
47. Yuan, K.; Zhang, K.; Zheng, Y.; Li, D.; Wang, Y.; Yang, Z. Irregular distribution of wind power prediction. *J. Mod. Power Syst. Clean Energy* **2018**, 6, 1172–1180, doi:10.1007/s40565-018-0446-9.
48. Bryc, W. *The Normal Distribution*, 1st ed.; Springer: New York, NY, USA, 1995; pp. 23–28.
49. El Dorado Weather. Alaska Mean Wind Speed and Prevailing Direction. 2020. Available online: <https://www.eldoradoweather.com/> (accessed on 14 November 2020).
50. Weather Spark. The Typical Weather Anywhere on Earth. 2020. Available online: <https://weatherspark.com/> (accessed on 14 November 2020).
51. SKYbrary. International Standard Atmosphere (ISA). 2017. Available online: [https://www.skybrary.aero/index.php/International\\_Standard\\_Atmosphere\\_\(ISA\)](https://www.skybrary.aero/index.php/International_Standard_Atmosphere_(ISA)) (accessed on 14 November 2020).
52. Steve, A. International Standard Atmosphere: How it Affects Flight—Understanding the Basics. 2014. Available online: <https://www.universalweather.com/blog/international-standard-atmosphere-how-it-affects-flight-understanding-the-basics/> (accessed on 14 November 2020).
53. U.S. Department of the Interior. What Are Digital Elevation Models (DEMs)? 2020. Available online: <https://www.usgs.gov/> (accessed on 14 November 2020).
54. NRC Services. National Elevation Dataset. 2020. Available online: <https://www.usda.gov/> (accessed on 14 November 2020).
55. U.S. Department of the Interior. The National Map. 2020. Available online: <https://www.usgs.gov/core-science-systems/national-geospatial-program/national-map> (accessed on 14 November 2020).
56. Arms, C.R.; Fleischhauer, C.; Murray, K. KML, Version 2.2. 2012. Available online: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000340.shtml> (accessed on 14 November 2020).
57. Schneider, A. GPS Visualizer: Do-It-Yourself Mapping. 2019. Available online: <https://www.gpsvisualizer.com/> (accessed on 14 November 2020).
58. Hazzard, C. What is a GPX File? 2020. Available online: <https://hikingguy.com/how-to-hike/what-is-a-gpx-file/> (accessed on 14 November 2020).
59. Thompson, W. *F-86 Sabre Fighter-Bomber Units Over Korea*; Osprey Publishing: Oxford, UK, 1999; pp. 8–73.
60. History.com Editors. Korean War. 2020. Available online: <https://www.history.com/topics/korea/korean-war> (accessed on 14 November 2020).
61. Samet, H.; Kochut, A. Octree approximation and compression methods. In Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission, Padova, Italy, 19–21 June 2002; pp. 460–469, doi:10.1109/TDPVT.2002.1024101.
62. Samaniego, F.; Sanchis, J.; Garcia-Nieto, S.; Simarro, R. Smooth 3D Path Planning by Means of Multiobjective Optimization for Fixed-Wing UAVs. *Electronics* **2019**, 9, 51, doi:10.3390/electronics9010051.
63. Oshana, R. 4—Overview of Digital Signal Processing Algorithms. In *DSP Software Development Techniques for Embedded and Real-Time Systems*; Oshana, R., Ed.; Embedded Technology, Newnes: Burlington, MA, USA, 2006; pp. 59–121, doi:10.1016/B978-075067759-2/50006-5.
64. Anjali, B.; Vivek, A.; Nandagopal, J. Simulation and analysis of integral LQR controller for inner control loop design of a fixed wing micro aerial vehicle (MAV). *Procedia Technol.* **2016**, 25, 76–83.
65. Boyd, S.; El Ghaoui, L.; Feron, E.; Balakrishnan, V. *Linear Matrix Inequalities in System and Control Theory*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1994; doi:10.1137/1.9781611970777.
66. Maciejowski, J. *Predictive Control: With Constraints*, 1st ed.; Prentice Hall: London, UK, 2002.
67. Cuzzola, F.A.; Geromel, J.C.; Morari, M. An Improved Approach for Constrained Robust Model Predictive Control. *Automatica* **2002**, 38, 1183–1189, doi:10.1016/S0005-1098(02)00012-2.
68. Maeder, U.; Borrelli, F.; Morari, M. Linear Offset-Free Model Predictive Control. *Automatica* **2009**, 45, 2214–2222, doi:10.1016/j.automatica.2009.06.005.
69. Mayne, D.; Rawlings, J.; Rao, C.; Scokaert, P. Constrained Model Predictive Control: Stability and Optimality. *Automatica* **2000**, 36, 789–814, doi:10.1016/S0005-1098(99)00214-9.
70. Mayne, D.Q. Model Predictive Control: Recent Developments and Future Promise. *Automatica* **2014**, 50, 2967–2986, doi:10.1016/j.automatica.2014.10.128.
71. Kouvaritakis, B.; Cannon, M. *Model Predictive Control. Classical, Robust and Stochastic*, 1st ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; doi:10.1007/978-3-319-24853-0.
72. Blanchini, F.; Miani, S. *Set-Theoretic Methods in Control*, 1st ed.; Birkhäuser: Boston, MA, USA, 2008; doi:10.1007/978-0-8176-4606-6.
73. Khlebnikov, M.V.; Polyak, B.T.; Kuntsevich, V.M. Optimization of Linear Systems Subject to Bounded Exogenous Disturbances: The Invariant Ellipsoid Technique. *Autom. Remote Control* **2011**, 72, 2227–2275, doi:10.1134/s0005117911110026.
74. Ortiz, Á. Autonomous Navigation for an F-86 Sabre. 2021. Available online: <https://es.mathworks.com/matlabcentral/fileexchange/85168-autonomous-navigation-system-for-an-f-86-sabre> (accessed on 14 November 2020).

75. Fleeman, E.L. *Tactical Missile Design*, 2nd ed.; American Institute of Aeronautics and Astronautics: Atlanta, GA, USA, 2006; pp. 21–80.
76. Blog, T.S.C. Coding Bilinear Interpolation. 2011. Available online: <http://supercomputingblog.com/graphics/coding-bilinear-interpolation/> (accessed on 14 November 2020).
77. Cutler, C. Ground Effect: Why You Float During Landing. 2020. Available online: <https://www.boldmethod.com/learn-to-fly/aerodynamics/what-happens-to-your-plane-in-ground-effect/> (accessed on 14 November 2020).
78. Airport Authorities India. What Is an ILS and Its Different Component? 2020. Available online: <https://www.aai.aero/en/content/what-ils-and-its-different-component> (accessed on 14 November 2020).
79. Demirhan, M.; Premachandra, C. Development of an Automated Camera-Based Drone Landing System. *IEEE Access* **2020**, *8*, 202111–202121, doi:10.1109/ACCESS.2020.3034948.
80. Bouhali, M.; Shamani, F.; Dahmane, Z.E.; Belaidi, A.; Nurmi, J. FPGA applications in unmanned aerial vehicles—A review. In *International Symposium on Applied Reconfigurable Computing*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 217–228.
81. Kunzler, J.W.; Ammermon, S.M.; Warnick, K.F. Progress Toward Airborne GPS Spatial Filtering Powered by Recent Advances in FPGA Technology. In *Proceedings of the 2020 Intermountain Engineering, Technology and Computing (IETC)*, Orem, UT, USA, 2–3 October 2020; pp. 1–6.