POLITECNICO DI TORINO
Repository ISTITUZIONALE

Formation Flight of Multiple UAVs Using Artificial Potential Field Algorithm

Original
Formation Flight of Multiple UAVs Using Artificial Potential Field Algorithm / Favia, A., Lerro, A., Gili, P., Papa, U., Chiesa, A.. - (2023). (AIAA Aviation and Aeronautics Forum and Exposition, AIAA AVIATION Forum 2023 San Diego, CA (USA) 12-16 June 2023) [10.2514/6.2023-4148].

Availability:
This version is available at: 11583/2997428 since: 2025-02-10T20:28:43Z

Publisher:
American Institute of Aeronautics and Astronautics, AIAA

Published
DOI:10.2514/6.2023-4148

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
AIAA preprint/submitted version e/o postprint/Author's Accepted Manuscript

(Article begins on next page)

27 November 2026

Formation flight of multiple UAVs using Artificial Potential Field Algorithm

Favia Alessandro *
Leonardo Company - Aircraft Division, Turin, Italy, 10146

Angelo Lerro † and Piero Gili ‡
Polytechnic University of Turin, Turin, Italy, 10129

Umberto Papa §
Leonardo Company - Aircraft Division, Pomigliano d'Arco, Naples, Italy, 80038

Alberto Chiesa ¶
Former Leonardo Company - Aircraft Division, Turin, Italy, 10146

In the present work, using a linearized model of UAV, a formation control algorithm capable of managing a formation of five aircraft has been designed. After studying the operating principle of the Artificial Potential Field in its static (not time-variant) version, a 2D Simulink model (which works only in the horizontal plane) has been created. This model has been subsequently extended to the vertical plane in order to obtain a formation control algorithm in three-dimensional space.

I. Nomenclature

APF	=	Artificial Potential Field
$MUM-T$	=	Manned-Unmanned Teaming
UAV	=	Unmanned Aerial Vehicle
NED	=	North-East-Down reference system
ECI	=	Earth-Centred-Inertial reference system
ABC	=	Aircraft-Body-Coordinates reference system
LTI	=	Linear Time-Invariant system
LQR	=	Linear Quadratic Regulator
PID	=	Proportional Integral Derivative

II. Introduction

In the near future *Unmanned Aerial Vehicles (UAVs)* will become increasingly important, especially for their great advantage of reducing risks regarding human presence in the field. The main issue with these platforms lies in the fact that their capabilities to operate outside of a military or segregated environment are quite limited. The achieved degree of autonomy is also limited and in fact most of them are remotely controlled or semi-autonomous, without the possibility to identify or handle out of the ordinary situations. The ability of unmanned aircraft to support manned ones during a mission is defined as Manned - Unmanned Teaming (MUM-T). The present work, which is the result of the collaboration between Politecnico di Torino and Leonardo Company (Aircraft Division), fits into this context. Using *Matlab/Simulink* a linear fixed-wing UAV model, that includes main flight control surfaces and autopilots, has been designed in order to simulate aircraft dynamics. As the leader-follower architecture has been adopted, a five-aircraft formation control

*Aerospace Engineer, Flight Mechanics and Control Laws, favia.alessandro@outlook.com
†Insert Job Title, Department Name, Address/Mail Stop, and AIAA Member Grade (if any) for first author.
‡Insert Job Title, Department Name, Address/Mail Stop, and AIAA Member Grade (if any) for first author.
§Insert Job Title, Department Name, Address/Mail Stop, and AIAA Member Grade (if any) for first author.
¶Insert Job Title, Department Name, Address/Mail Stop, and AIAA Member Grade (if any) for first author.

algorithm based on PID (*Proportional - Integral - Derivative*) controllers has been implemented. PID is a possible automatic control that enables statically controlled formations. Thus, the autonomy of the unmanned platforms is not accounted for and the safe feasibility of the controlled formation is not guaranteed. For this reason it was decided to change the approach, generating a system based on *Artificial Potential Fields (APF)*. Artificial potentials allow the overlapping of different constraints which should be considered when dealing with formations (e.g. to perform a specific manoeuvre, to maintain a safe distance from the other agents), all weighted through a cost function. The "desired" position of the UAV is commanded, but the UAV is left with the autonomy to implement the command basing on needs. This is a crucial aspect both in terms of the use of *self-awareness* of the platform and its *decision-making* capabilities. After studying the operating principle of the algorithm in its static (not time-variant) version, a two-dimensional Simulink model (which works only in the horizontal plane) has been created. This model has been then extended to the vertical plane in order to obtain a formation control algorithm in a three-dimensional space. In both 2D and 3D cases the code is able to perform manoeuvres and formation geometry variations in real time and it can be controlled by the user with direct (run time) inputs. Along with the capability to avoid collision with both obstacles and formation components, any follower aircraft can also avoid flying over no-fly zones. The proposed solution for formation flight control can can be easily adapted to the selected type of aircraft and to mission needs by tuning specific parameters. Trajectories, attitude and potentials visualizations are carried out using both real-time and post-production animations.

III. Formation control and motion planning of multiple UAVs: the state of the art

The reasons for studying how a number of aircraft could fly closely and autonomously are numerous. First of all, it has been demonstrated that flying in the wing-tip vortex of an aircraft exhibit fuel saving [1] [2]. Particularly useful during long range missions, it can be explained observing the reduction of induced drag of the trailing wingmen, which need less energy to maintain its speed [3]. This behaviour has been observed in many species of birds, especially the migratory ones, which travel in "V" formation during long journeys. Another reason for designing robust formation flight algorithms lies in using them in the civil sector, with the urgent demand for aircraft coordination within high density airspace, especially near airports [4]. *Autonomous Aerial Refuelling (AEF)* is another research area of interest, both in civil and military sector [5] [6]. From the military point of view, the interaction between multiple *Unmanned Aerial Vehicle* and (possibly) manned aircraft is a very important issue. As can be seen in [7] the *Ghost Bat* program of the Royal Australian Air Force (RAAF), for example, aim to develop a loyal-wingman system together with Boeing.Another increasing trend is the development of more sophisticated *unmanned vehicles*, which can operate in multiple mission scenarios. The reduction of risks regarding the human presence on the field is only one of the reasons why these systems are becoming so common. The degree of autonomy reached today does not allow them to act completely independently, so they are generally *semi-autonomous* because the human intervention is needed to assure a reasonable reliability. In this context, adopting a *fleet* of UAVs instead of a single one could be an interesting solution since it can assure an increase in mission capability as well as augmented redundancy.
Acoording to [8], two main research areas particularly prolific: *formation control* and *cooperative motion planning*.

A. Formation control architecture

The first formation control approach is called *"leader-follower"*. An aircraft is elected as *leader*, it has all the information about the trajectory to follow to reach a certain point in the space and its motion define the one of all the others, called *"followers"*. Their purpose is to maintain a certain distance from the *leader* geometrically calculated from the desired formation shape. This approach has been extensively studied in literature, with numerous variations on the theme. In [9], every element (in this case, robots) is considered as point-mass and the control law aim to nullify distance and angular errors then the desired values. A similar strategy can be noticed, for example, in [10]. In other cases collision-avoidance features can be implemented, as in [11] and [12]. The main problem of this approach is that there must be a good quality communication between all the aircraft (especially with the leader) in order to transmit position data. These uncertainties could be partially modelled using, among others, machine-learning techniques.

The second formation control approach is called *"virtual structure"* [13]. In this case there is no distinction between aircraft, which are equally important. The aim of every vehicle is to minimize the position error with respect to a vertex of the virtual structure (the formation *shape*), treated as a rigid body. The virtual structure is translated to follow a predetermined trajectory, forcing every aircraft to move to the subsequent vertex position. As compared to the previous strategy, one of the advantages is the reliability in generating and maintaining the shape, but the major drawback is that collision avoidance could be particularly difficult to implement.

The third formation control approach is the *"behaviour-based"* one [14]. The formation control problem is

solved using a *hybrid vector-weighted control function*. In particular, different control schemes (e.g. *move-to-goal* or *avoid-static-obstacle*) are developed and the general control scheme for the specific mission is the result of the weighting of them (using gain values). The main problem of this approach is that the above-mentioned general control scheme does not depend on kinematic/dynamic characteristics of the agents, so the mathematical study of system stability become quite complex to pursue and this makes it hard to theoretically justify the performance of this approach [15].

B. Features of the proposed approach

During the initial phase of requirement definition, some important mandatory features of the novel algorithm proposed in the present work came to light. First, the code has to consider the presence of a *manned aircraft*, followed by a number of *unmanned aircraft*. Every UAV must follow the *manned* aircraft autonomously in order to generate a predefined formation pattern. Contacts between vehicles must be avoided, and *collision avoidance* should be implemented to avoid fixed or mobile obstacles. According to preliminary design analysis, the leader-follower approach emerged as the most promising for the mission requirements and the presence of a manned leader. Relatively simple to design, this approach has another important advantage, which is the *centralized communication architecture*. For example, all data needed to carry out the separation between aircraft result in a direct link between the *leader* and the single *follower*, with a reduction in the overall quantity of exchanged data as compared to a *decentralized communication architecture*.

C. The cooperative formation path planning

Cooperative formation path planning is a field of study which works in parallel with *formation control*. In general, path planning consists in finding an optimal or near-optimal path from the starting state to the target state that avoids obstacles based on one or some performance indicators. For a 3-D path planning algorithm, for example, the problem could be schematized as:

$$P_s(x_s, y_s, z_s, \psi_s, \theta_s, \varphi_s) \overset{r(t)}{\rightarrow} P_f(x_f, y_f, z_f, \psi_f, \theta_f, \varphi_f) \tag{1}$$

where P_s and P_f are the starting and final configuration (defined by positions x, y, z and attitude angles ψ, θ, φ) and $r(t)$ is the computed path.

Consider a generic agent (be it a robot, a UAV etc.). The path choice, for a single-agent case, depends on various factors such as lowest working cost, shortest walking route and shortest walking time. Extending the problem for a formation of N agents, a number of N paths have to be calculated taking into account additional performance costs [8] like:

- *Internal collision avoidance*: vehicles should avoid others inside the formation
- *Formation behaviour*: if a formation pattern should be maintained, paths are influenced
- *Cooperation behaviour*: there are two different forms of cooperation behaviour, *time cooperative behaviour* and *time and position cooperative behaviour*. The main difference lies on the fact that the formation could be broken in encountering an obstacle (first case) or it should be always maintained (second case).
- *Total distance*: the path optimization should take into consideration not only the single N-th path length, but also the sum of distances.

There are several ways to categorise path-planning algorithms. For example, they could be divided according to the know degree of environmental information: global-map based (*global path planning*) or local-map based (*local path planning*) [16]. In the first case it is assumed that the environment is completely known, especially the obstacle position and shape: an environmental model is created (using, for example, a grid decomposition) and the path is calculated. In the second case the surrounding is sensed only using *sensors*, so the path is based on limited perception of the environment and it should be adjusted in real time.

Another classification refers to the *deterministic* or *heuristic* approach. In the deterministic approach, the solution for the trajectory exists and it is exact (being made up of a finite number of step): if nothing changes, the output of the simulation remains the same no matter how many times the same is run. It can be considered *complete* and *consistent*. Conversely, a heuristic approach is used if an exact solution does not exist or if it is difficult to find. In this case the solution is *near-optimal* since it is the result of an approximation and it generally change if the simulation is repeated.

D. Formation path planning algorithms

According to [8], the currently most used classes of algorithms for control of multiple UAV are *Artificial Potential Field (APF)*, *optimal control*-based methods and *evolutionary algorithms*.

The *Artificial Potential Field* is a powerful path-planning algorithm that was first proposed by Oussama Khatib in 1986. Generally speaking this method computes, in every point in which the physical space has been discretized, a potential field. There, an *attractive* potential is centred in the target point and influences all the surrounding space while *repulsive* potentials are centred in the obstacles and act only in certain areas around them. The *steepest descent* direction can be calculated using the gradient of the potential which, in this case, corresponds to a *force*:

$$F_{total} = F_{att} + F_{rep} = -\nabla(U_{att} + U_{rep}) \tag{2}$$

When used for formation path planning other type of potential should be implemented, in order to avoid collision between robots and maintain formation shape: Fig. 1 is an example of a potential where both attractive and repulsive terms are included. A particular version of APF is called *"Fast Marching Method based Potential Field"* [17]. Being the result of the application of the Voronoi Fast Marching (VFM) method to the APF, it generates the potential field simulating the propagation of an electromagnetic wave (a *viscosity map* is extracted from the discretized space grid). In this case the algorithm is faster, even though more complicated to implement.

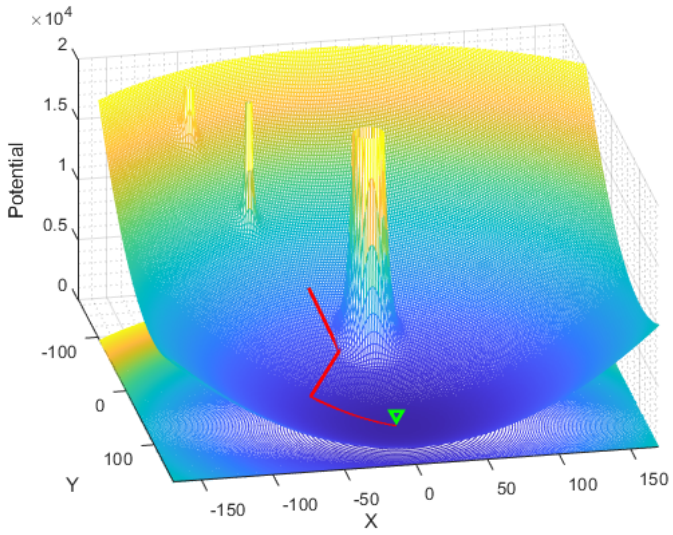


Fig. 1 Example of artificial potential field

The *optimal control method* represents another approach to the formation path planning. In this case, the problem is faced dividing it into several sub-problems (one for each vehicle), solved using numerical optimization and considering a set of constraints. A basic formulation is available in [18]. Let denote the initial state of the vehicle with $\boldsymbol{x}_{init}$ and the desired (final) state with $\boldsymbol{x}_f$. The planning horizon of the algorithm T (i.e. how far in time the route is calculated) depends on a number of factors, like the distance over which the space has been discretized. The *cost function* for the i^{th} time step is $\ell_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_f)$ where $\boldsymbol{u}_i$ is the input vector. The problem can be formulated as follow:

$$\min_{\boldsymbol{x}_i, \boldsymbol{u}_i} J_T = \sum_{i=0}^{T-1} \ell_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_f) + f(\boldsymbol{x}_T, \boldsymbol{x}_f) \tag{3}$$

$$\begin{aligned} \text{subject to: } & \boldsymbol{x}_{i+1} = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{B}\boldsymbol{u}_i \\ & \boldsymbol{x}_0 = \boldsymbol{x}_{init} \\ & \boldsymbol{x}_i \in \mathcal{X} \\ & \boldsymbol{u}_i \in \mathcal{U} \\ & (x_i, y_i) \in \mathcal{D} \\ & (x_i, y_i) \notin \mathcal{O} \end{aligned}$$

where $(\boldsymbol{A}, \boldsymbol{B})$ represent the linear dynamics of the vehicle, (x_i, y_i) denotes the position of the vehicle in the plane, the set $\mathcal{D}$ represents the discretized physical space and the set $\mathcal{O}$ characterizes the obstacles. Sets $\mathcal{X}$ and $\mathcal{U}$ represent the dynamic and kinematic constraints of the vehicle, such as maximum turn rate and minimum speed restrictions. Using, for example, *Mixed Integer Linear Programming (MILP)*, the problem could be solved. The main disadvantage of this method is the solver high computation complexity, which could make it difficult to use it for on-line applications.

The *evolutionary algorithm (EA)* is a *heuristic* algorithm inspired to biological evolution process. In case of path planning, it mimics the natural selection considering every path as an individual which mutate with time. Using a number of constraints and selection criteria the best path is chosen for every vehicle. When used for the cooperative path planning, it generally consists of two processes: a *N-th vehicle process* calculate the optimal trajectory for every single vehicle separately while the *master process* takes into consideration the cooperative behaviour. In particular, it re-evolves every path considering collision avoidance and distances inside the formation.

When a single vehicle is involved in the simulation, grid and map-based algorithms are preferred, like *D-star* [19] and *rapidly-exploring random trees* [20] respectively. For formation path planning, where the computational time is extremely important (algorithms become more and more complicated with the increase in the number of the vehicles), APF and Evolutionary Algorithms are the first choice. In addition to be faster than others, they produce smooth trajectories which can be easily followed by the UAVs. For the purpose of this paper a "traditional" *Artificial Potential Field* has been chosen for its simplicity and effectiveness. It has been extensively used for unmanned aerial vehicles and it can be easily adapted to various situations.

IV. The aircraft: general characteristics and performances

In order to create a virtual test environment for our guidance and control algorithms, it is necessary to choose an appropriate plant model. In general, a formation could include various type of aircraft but in this dissertation, for the sake of simplicity, only a single type will be considered. The aircraft in question is a *medium-altitude long-endurance Unmanned Aerial Vehicle*: the main characteristic of this UAV class is their capacity to fly at an altitude between 10000 ft and 30000 ft (3048 − 9144 m) for about 24 − 48 hours. Examples of this type of aircraft are the *MQ-1 Predator* produced by *General Atomics* and the *Falco Xplorer* produced by *Leonardo*.
Plant dynamics is linearised and has been simulated using the *state-space approach*. Generally speaking, the state variables of a system are a minimum set of variables $x_n(t)$ that, when known at time t_0 and along with the input, are sufficient to determine the state of the system at any other time $t > t_0$. Linearized aircraft equation of motion [21] can be written in state-space form:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\eta} \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{\eta} \end{cases} \tag{4}$$

where a) $\boldsymbol{x}$ is the state vector; b) $\boldsymbol{\eta}$ is the control vector; c) $\boldsymbol{y}$ is the output vector; d) $\boldsymbol{A}$ is the plant matrix; e) $\boldsymbol{B}$ is the input matrix; f) $\boldsymbol{C}$ and $\boldsymbol{D}$ are, in our case, an identity and a null matrix respectively.

If $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ and $\boldsymbol{\eta}$ are known, we can describe how the aircraft behaves over time. It should be specified that all the matrices depend on altitude, centre of gravity and velocity, but for the purposes of our work the system is time-invariant. Moreover, $\boldsymbol{x}$, $\boldsymbol{\eta}$ and $\boldsymbol{y}$ are defined as variation with respect to the trim conditions: it means that the state vector consist of

V_{trim}	H_{trim}	α_{trim}	θ_{trim}	$i_{H,trim}$	$\delta_{A,trim}$	$\delta_{R,trim}$	T_{trim}
$(\frac{m}{s})$	(ft)	(deg)	(deg)	(deg)	(deg)	(deg)	(kg)
82.3111	10000	$\simeq 3.5$	$\simeq 3.5$	$\simeq 0.5$	0	0	$\simeq 600$

Table 1 UAV trim conditions

increments $\Delta x_1, \Delta x_2 ..., \Delta x_n$ (where n is the *order of the system*). Every matrix has been provided by Leonardo (Aircraft Division) but, for reasons of company intellectual properties, they will not be described here.
For the present work, the longitudinal motion vectors are:

$$\boldsymbol{x}_{lon} = \begin{bmatrix} V \\ \alpha \\ q \\ \theta \end{bmatrix} = \boldsymbol{y}_{lon} \quad \boldsymbol{\eta}_{lon} = \begin{bmatrix} \delta_E \\ \delta_T \end{bmatrix} \tag{5}$$

with δ_E = *elevators deflection* and δ_T = *throttle* ($\boldsymbol{C} = \boldsymbol{I}$). In the same way, for the lateral-directional motion:

$$\boldsymbol{x}_{latdir} = \begin{bmatrix} \beta \\ p \\ r \\ \varphi \\ \psi \end{bmatrix} = \boldsymbol{y}_{latdir} \quad \boldsymbol{\eta}_{latdir} = \begin{bmatrix} \delta_A \\ \delta_R \end{bmatrix} \tag{6}$$

where δ_A = *ailerons deflection* and δ_R = *rudder deflection*. For the purpose of this dissertation trim conditions are considered known (Tab. 1).

A. The Simulink model of the aircraft

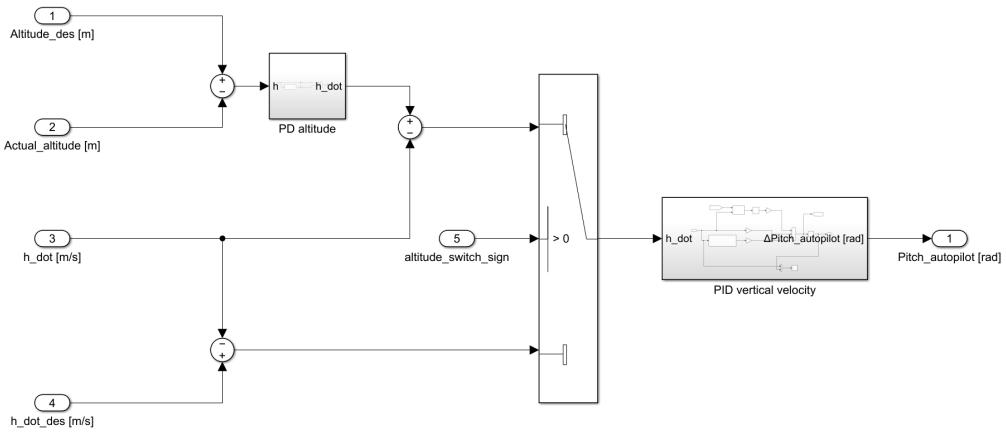
Once the aircraft has been chosen, the following step consists in trying to simulate its dynamic behaviour. The cooperative formation control system generates high - level commands which guarantees the separation between leader, followers and obstacles. The nature of these commands and the type of variable which APF controls could be different on a case-to-case basis and their choice is made during the design phase. However, the output that the formation control system generates must be converted in variables which the state-space system can accept.

From the Simulink model point of view, the plant has been divided in three parts:

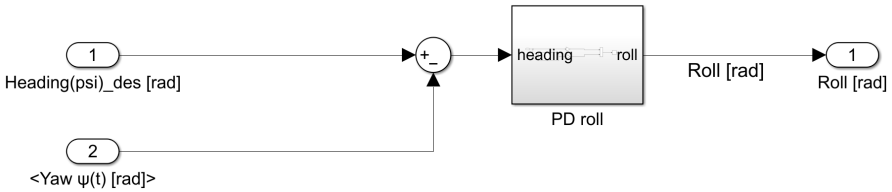
- The aircraft linearized dynamic equations
- Primary control laws & autothrottle
- Autopilots

The *aircraft linearized dynamic equations* block describes the dynamic of the aircraft using the state space approach described in the previous section (Eq. 4). Longitudinal and lateral-directional motion can be separated thanks to the *small-perturbations* hypothesis applied to the equations. *Navigation equations* are employed in order to calculate positions and velocities in NED frame. Inputs of the previous block are essentially the aircraft controls, that are throttle and deflections of mobile surfaces. Such deflections result from a control system that calculates them based on a "desired" that comes, in our case, from the *Autopilots* block. It is therefore necessary to design these control laws, building the block *Primary control laws & autothrottle*. The inputs of this subsystem are essentially two: on the one hand there are the autopilot control and on the other the plant data, useful for the calculation of incoming errors from the various control algorithms; as we will see, only for the leader aircraft the autopilot command can be substituted with the *joystick* ones, so that the user can directly control its trajectory.

The longitudinal motion has been controlled using two different *Linear Quadratic Regulator* (LQR) [22], one for the elevators and one for the throttle. The separation of the control loops acting on the same dynamic model is advisable because of the very weak correlation between them, so there is no real benefit in optimizing them by building an overall control law. In order to tune $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices, a Simulink model has been created in which the plant is subjected to a



(a) Altitude/Vertical velocity autopilot



(b) Vertical velocity

Fig. 2 Autopilots block schemes

Step signal. The response of the system over time can be observed and modified according to the defined requirements. The *throttle* is controlled by means of a ΔV required by the *Autopilot* block.

As far as lateral-directional control is concerned, the design procedure is slightly different from the previous one. The *rudder* deflection causes an unwanted roll motion, which must be corrected. On the other hand, *aileron* deflection causes an unwanted yaw motion, which must be also corrected. Designing the ailerons and rudder LQRs separately this effect would have not been considered, making the aircraft unstable. The adopted solution is to use a control system that synergistically involves both mobile surfaces. The LQR state vector $\widetilde{\boldsymbol{x}}_{latdir}$ is composed by all the lateral-directional state variables to which $\int \beta$ and $\int \varphi$ have been added. Gain matrices $\boldsymbol{K}_{LQR,A}$ and $\boldsymbol{K}_{LQR,R}$ have been calculated using a single *lqr* function. It should be noted that, for reasons due to the *Autopilot* block design, the directional motion is controlled through the sideslip angle β (instead of the yaw angle ψ).

The input of the *"Flight surfaces control and autothrottle"* block is a vector composed by four elements:

$$AP_{out} = \begin{bmatrix} \Delta\theta_{AP_out} & \Delta\varphi_{AP_out} & \Delta\beta_{AP_out} & \Delta V_{AP_out} \end{bmatrix} \tag{7}$$

This vector includes the variations of all the quantities that the aircraft needs to perform a certain manoeuvre, which are produced by the *"Autopilots"* subsystem. There are various autopilots in literature whose implementation depends, among other things, by the technological level of the vehicle. In our case we have chosen to implement:

- *"Altitude/Vertical velocity autopilot"*
- *"Heading autopilot"*

The outputs of this control blocks are, respectively, $\Delta\theta_{AP_out}$ and $\Delta\varphi_{AP_out}$. The β_{AP_out} has been set to zero because it has been considered that the aircraft can perform only *coordinated turns*. It has not been necessary to implement an additional algorithm for the velocity because the control logic for the manoeuvres produces a ΔV that can be directly used in the *throttle LQR controller*. Rather than using the autopilots the leader aircraft can be directly controlled by the user.

The *Altitude/Vertical velocity autopilot* is represented in Fig. 2a and is composed of a "PD altitude" controller and a "PID vertical velocity" controller. Outputs of both controllers are, respectively, $\Delta\dot{h}_{des}$ and $\Delta\theta_{des}$.

Once the autopilots were designed, it was necessary to build a control logic that would allow them to be used. The necessary data requested by autopilots are:

- h_{des}

- $\dot{h}_{des}$
- $altitude_switch_sign$
- $heading_{des}$
- β_{des}
- V_{des}

Using a *Switch* function, nine different *"Fundamental flight modes"* can be selected:

- Reach the desired altitude
- Maintain the desired altitude
- Reach the desired vertical velocity
- Reach the desired heading angle with constant altitude
- Reach the desired heading angle with variable altitude
- β changing with constant altitude
- β changing with variable altitude
- Reach the desired velocity with constant altitude
- Reach the desired velocity with variable altitude

Every follower in the formation can operate in two ways:

- *"Autonomous flight" mode*: the follower aircraft execute the manoeuvre independently. The formation control algorithm is turned off and collision-avoidance is not guaranteed.
- *"Formation flight" mode*: the follower aircraft flies in formation. This position can be user-selected and different *formation shapes* can be built.

V. Formation control using Proportional Integral Derivative controllers

In Chapter IV the way in which the model of the aircraft and the logic for the manoeuvres have been designed have been described. Combining the *"Fundamental flight modes"*, every aircraft is completely independent from each other, that is it follows only user commands without sensing other followers. The next step is to create a simple cooperative formation control algorithm which allows the followers to maintain a relative position with respect to the leader. In this initial development phase, the *collision avoidance* is not considered. After examining the state of the art of formation control and motion planning and before facing the problem of the *Artificial Potential Field*, it has been decided to study how a PID-based formation algorithm can be used in a realistic application.

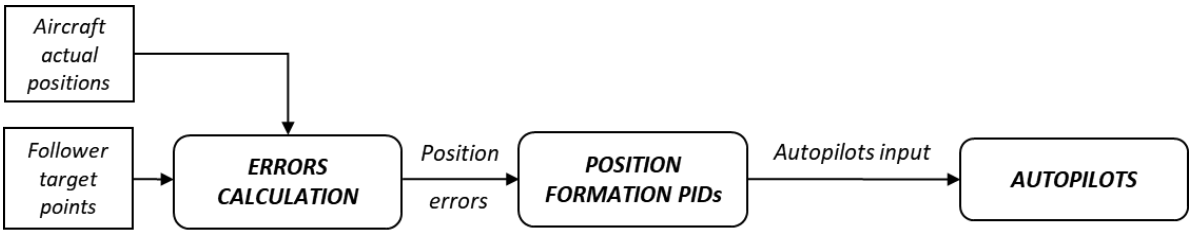


Fig. 3 Formation flight PID controllers

A block scheme of the proposed algorithm is depicted in Fig. 3. As suggested in [23] the formation control can be decomposed in two different problems, namely calculation of position errors on horizontal and vertical planes. First of all, the inertial NED reference system should be identified. In our case and for how the Simulink model was structured, this system is defined for $t = 0\ s$ with the X axis pointing north, the Y axis pointing east and the Z axis pointing down. PN_0, PE_0 e PD_0 (i.e. initial positions) of leader and followers are all defined with respect to the NED system so they are absolute distances. Followers must occupy a certain position relative to the leader, which can vary with the mission. This position, seen as a vector in a *leader-centred reference frame*, can be defined using its three components:

- *Forward clearance* f_c
- *Lateral clearance* l_c
- *Height clearance* h_c

The main purpose of the PID-based formation control is to drive the followers toward their desired positions. These PIDs needs *position errors* in a follower-centred system, errors that we have to calculate. PN, PE and PD derive from the plant block and are in NED frame. The relative altitude Δh_{LF} can be easily computed as a difference between h_L

and h_F while for the *altitude error* the height clearance h_c should be added:

$$h_{formation\,error} = \Delta h_{LF} + h_c + [-1 \cdot (PD_{0,F} - PD_{0,L})] \tag{8}$$

The term $(PD_{0,F} - PD_{0,L})$ is introduced because if the initial condition coincides with the target point the formation error should be zero. Forward and lateral formation errors are treated differently and heading angles ψ_L and ψ_F should be introduced. Being l_c and f_c known in a *leader reference frame*, these distances should be transformed with respect to the follower:

$$\begin{bmatrix} f_{c,F} \\ l_{c,F} \end{bmatrix} = \begin{bmatrix} cos(\psi_F - \psi_L) & sin(\psi_F - \psi_L) \\ -sin(\psi_F - \psi_L) & cos(\psi_F - \psi_L) \end{bmatrix} \begin{bmatrix} f_c \\ l_c \end{bmatrix} - \begin{bmatrix} PN_{0,F} \\ PE_{0,F} \end{bmatrix} \tag{9}$$

The corresponding formation error are:

$$\begin{bmatrix} f_{formation\,error} \\ l_{formation\,error} \end{bmatrix} = \begin{bmatrix} cos(\psi_F) & sin(\psi_F) \\ -sin(\psi_F) & cos(\psi_F) \end{bmatrix} \begin{bmatrix} \Delta PN + (PN_{0,F} - PN_{0,L}) + f_{c,F} \\ \Delta PE + (PE_{0,F} - PE_{0,L}) + l_{c,F} \end{bmatrix} \tag{10}$$

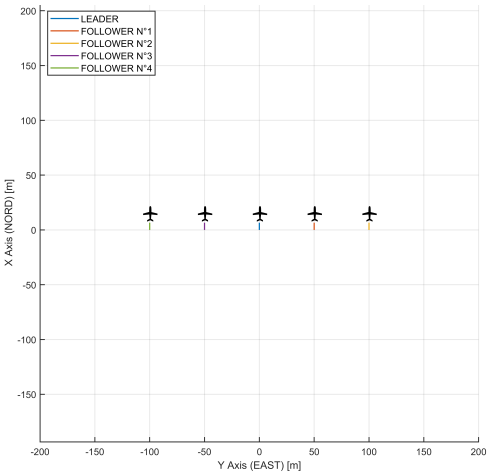
The reason for using heading angles is that the X axis orientation of the follower (or leader)-centred reference frame could not be parallel to X_{NED} at $t = 0\,s$ and it varies during turns.

A. Formation PID tuning

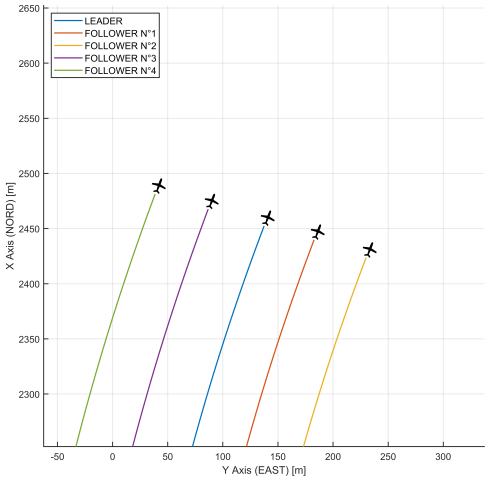
It is possible to introduce the formation PID controllers and their gains, starting from the *"Lateral formation PD (position)"* block. It is a proportional-derivative control (it was not necessary the use of the integrative line) which, receiving in input the $l_{formation\,error}$ generates a $\Delta\psi$. As others control blocks in this dissertation, gains have been obtained through a *trial-and-error* approach, imposing a step-like lateral clearance l_c and observing how the error tends to zero. The output has been limited in order to avoid excessive trajectory deviations. The *Forward formation PD (position)* block is similar to the previous one, not having the Integrator line. In this case the velocity saturation has been inserted in the *Throttle LQR* in order to limit both $manoeuvre$ and PD signals. The *Height (dot) formation PD (position)* block is basically identical to the other PDs. In this case the saturation is identical to the *PD altitude* autopilot one, avoiding $\dot{h}$ values not included in $\pm 1000\,\frac{ft}{min}$ ($\pm 5.08\,\frac{m}{s}$).

B. Simulation

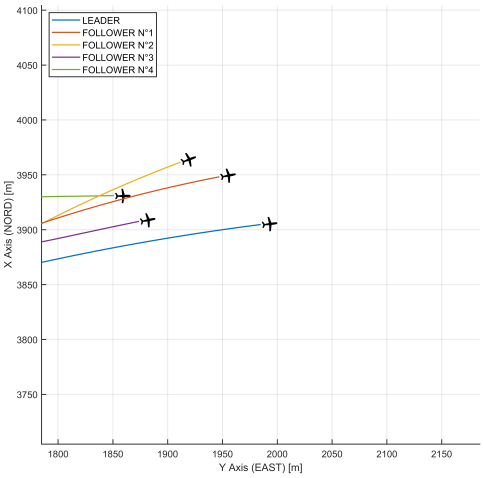
For the first twenty seconds every aircraft remains in its initial position (*line abreast* figure) with respect to the leader, with a straight trajectory without any change in altitude (Fig. 4a). Formation errors are equal to zero and PIDs do not intervene. At $t = 20\,s$ a right turn is required ($\psi_{des} = 180\,deg$) with altitude increase ($\dot{h}_{des} = 2\,\frac{m}{s}$), as can be seen in Fig. 4b. During this turn followers are ordered to change the geometry of the formation in a *line astern* type. It can be observed that aircraft initially closer to the centre of curvature (followers number one and two) have more difficulty reaching their target point (Fig. 4c). This is probably due to the fact that they have to turn in the opposite direction of the manoeuvre and successively compensate for the roll overshoot. At $t = 100\,s$ another formation patten is ordered, that is the *echelon*: in this case every aircraft manages to reach its position quite easily (Fig. 4e).
It can be observed that every aircraft follows the leader from a qualitative standpoint and places itself the formation. No space grids have to be handled (unlike the APF case) while the main problem is the complete lack of *collision avoidance*.



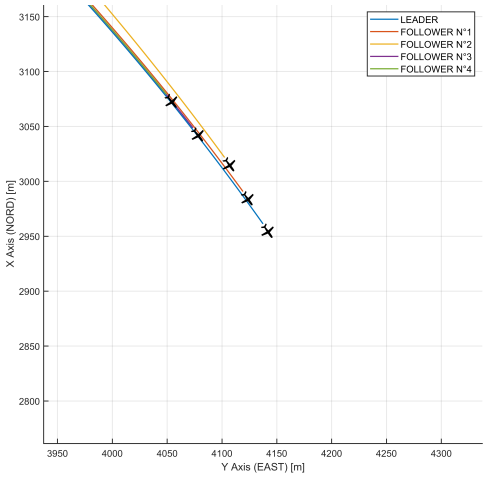
(a) 2D view (XY) - $t = 0\,s$



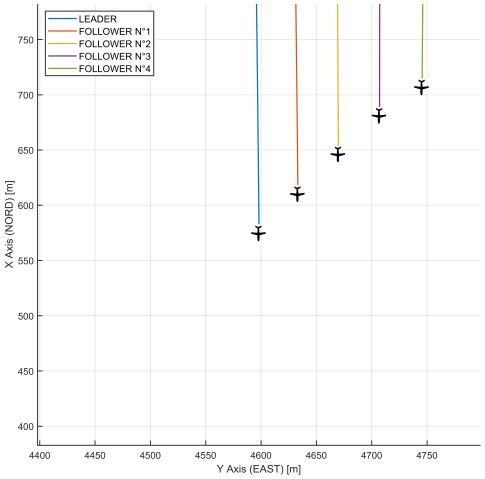
(b) 2D view (XY) - $t = 30\,s$



(c) 2D view (XY) - $t = 60\,s$



(d) 2D view (XY) - $t = 90\,s$



(e) 2D view (XY) - $t = 120\,s$

Fig. 4 2D views of the PID Formation control algorithm simulation

VI. Artificial Potential Field (APF) algorithm fundamentals

A. The traditional APF approach

As mentioned before, the PID-based formation control algorithm described in the previous section does not include *collision avoidance* feature as obstacles can not be sensed by the code itself. It was necessary to look for a new solution and, observing studies currently available in literature, it was clear that the *Artificial Potential Field (APF) method* could be an elegant and efficient solution to the *on-line* motion planning problem. Basically, the point $\boldsymbol{q}$ that represents the robot in the *operational space* moves under the influence of a *potential field* U, sum of *attractive* and *repulsive* terms. Multiple potential functions are available in literature whose choice depends on the nature of the problem. The attractive potential must be a concave function with a minimum in $\boldsymbol{q}_{goal}$ while the repulsive potential must be a non-negative, continuous and differentiable function whose value should tend to infinity as we approach the obstacle. In each point, an *artificial force* can be calculated as $-\nabla U(\boldsymbol{q})$, whose direction is the *steepest descent direction* toward $\boldsymbol{q}_{goal}$. The first appearance of this type of algorithm can be observed in 1986 in the paper *"Real-Time Obstacle Avoidance for Manipulators and Mobile Robots"* by Oussama Khatib [24] while different variations on the theme have been published until today.

The ***total potential*** $U_{tot}(\boldsymbol{q})$ has been defined as the sum of attractive and repulsive components [25]:

$$U_{tot}(\boldsymbol{q}) = U_a(\boldsymbol{q}) + U_r(\boldsymbol{q}) \tag{11}$$

while the *total force* $F_{tot}(\boldsymbol{q})$, considering a unique $\boldsymbol{q}_{goal}$ and p obstacles, is:

$$F_{tot}(\boldsymbol{q}) = -\nabla U(\boldsymbol{q}) = F_a(\boldsymbol{q}) + \sum_{i=1}^{p} F_{r,i}(\boldsymbol{q}) \tag{12}$$

The ***attractive potential*** can be both *parabolodic* and *conical*. In our application the first solution has been selected, that is the one illustrated in [24]:

$$U_a(\boldsymbol{q}) = \frac{1}{2} K_a \|e(\boldsymbol{q})\|^2 \tag{13}$$

where $e(\boldsymbol{q}) = \boldsymbol{q}_g - \boldsymbol{q} = \boldsymbol{x}_g - \boldsymbol{x}$ ($\boldsymbol{x} = [x, y, z]^T$). The term $K_a > 0$ should be tuned in order to force the robot to reach the target point.

The ***repulsive potential*** is added in order to consider obstacles with the idea of building a potential barrier around them, considering both their physical dimension (R_{obs}) and a specific tolerance ($toll$). In this case the function is of *piecewise-type* because we want the repulsive potential to act inside an established *range of influence* $\eta_{0,i}$. In general, $U_r(\boldsymbol{q})$ should be non-negative, continuous and differentiable, tending to infinity as we approach the obstacle. Khatib [24] proposed (*FIRAS function*):

$$U_{r,i}(\boldsymbol{q}) = \begin{cases} \frac{1}{2} K_{r,i} \left(\frac{1}{\eta_i(\boldsymbol{q})} - \frac{1}{\eta_{0,i}}\right)^2 & \text{if } \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ 0 & \text{if } \eta_i(\boldsymbol{q}) > \eta_{0,i} \end{cases} \tag{14}$$

where $\eta_{0,i} = R_{obs} + toll$, $\eta_i(\boldsymbol{q}) = |\boldsymbol{x} - \boldsymbol{x}_{obs}|$ and $K_{r,i} > 0$ is a repulsive constant. In our dissertation, we will use the repulsive function proposed by Dang et al. [12]:

$$U_r(\boldsymbol{q}) = \frac{1}{2} K_{r1} \left(\frac{r_0 + K_{r2}}{\|d_{obs}\| + K_{r2}} - 1 \right)^2 \tag{15}$$

because it depends on two constants, K_{r1} and K_{r2}, which make it more flexible to tune. $r_0 = (R_{obs} + toll)$ is the *repulsive radius* and d_{obs} is the difference between the point of the grid $\boldsymbol{x}$ and $\boldsymbol{x}_{obs}$:

$$\boldsymbol{d_{obs}} = \boldsymbol{x} - \boldsymbol{x}_{obs} \tag{16}$$

An important issue of potential function is the existence of *local minima*, i.e. places where $F_{tot} = 0$. In this case the algorithm could produce an error, because the robot is trapped there. As specified in [25] if every obstacle is modelled as a *sphere* (as in our case) and no zero-curvature obstacles exist, local minima do not constitute a threat.

B. APF for cooperative formation control

The approach described in Section VI.A has been called "traditional" because it considers a static environment with a single robot. In the present work, multiple agents are considered in order to generate a formation. Every robot must not collide with the others, and it must arrange itself in a predefined position following the leader. Finally, if obstacles exist, they must be avoided. As can be imagined, cooperative formation avoidance requires a different approach instead of the traditional one. In our case the potential of every follower is similar to [12], but the attractive part of U_i^j does not exists. As it will explained later, potentials will be built in a *leader-centred* frame and their components will be:

- *Inter-vehicle potential* U_i^j: repulsive potentials *only*, to avoid collisions between aircraft
- *obstacles potential* U_i^o: used for obstacle avoidance
- *target point potential* U_i^l: used to force the follower to arrange in the specific position inside the formation.

The APF used in the present work is defined *"pairwise"* because the potential for each agent is defined on the base of its state compared to that of one or some of the others [26].

Once the potential has been created, a feasible command for the aircraft (e.g. reference trajectory or direction) should be generated. Actually, the way this problem should be faced critically depends on its nature. In a static well-known environment (*off-line motion planning* problem), the trajectory could be computed once and then followed by the robot. It could be the case of a robotic arm which operate in a isolated chamber, without and other moving object. If the environment changes with time, computing the entire trajectory could be useless because after a while an obstacle could interfere with the same trajectory. In both situations, however, the most used approach is the *gradient descent algorithm*. The basic idea is quite simple: starting at the initial configuration, the gradient is computed and a step in its the opposite direction is taken. The process is repeated until we reach the target point, which is where $U = 0$. The latter condition condition is often replaced with $\|\nabla U(q(i))\| < \epsilon$ where ϵ is a sufficiently small tolerance. According to [27], the algorithm can be summarized as in Algorithm 1. The notation $q(i)$ is used to denote the value of q at i-th iteration while the path consists in the sum of segments generated by $\{q(0), ..., q(i)\}$. The term $\alpha(i)$ is the *step size* and determines the length of every segment. It can be both constant and variable value, depending upon mission requirements.

Algorithm 1 Gradient Descent

Input: A means to compute the gradient $\nabla U(q(i))$ at a point q (e.g. APF)
Output: A sequence of points $\{q(0), ..., q(i)\} \rightarrow$ *route*

1: $q(0) = q_{start}$
2: $i = 0$
3: **while** $\nabla U(q(i)) \neq 0$ **do**
4: $\quad q(i+1) = q(i) + \alpha(i)\nabla U(q(i))$
5: $\quad i = i + 1$
6: **end while**

When the solution to the problem has been found, we need to transform it in something that can be interpreted by the plant. According to [25], using the force $F_{tot}(q)$, three *planning techniques* exist:

1) The force $F_{tot}(q)$ can be considered as a vector of *generalized forces* τ, inducing a motion which depends on the plant dynamics:
$$\tau = F_{tot}(q) \tag{17}$$
2) The force $F_{tot}(q)$ can be considered as a vector of *generalized accelerations* $\ddot{q}$. The effect of the robot, seen as a point mass, is independent on its dynamics:
$$\ddot{q} = F_{tot}(q) \tag{18}$$
3) The force $F_{tot}(q)$ can be considered as a vector of *generalized velocities* $\dot{q}$:
$$\dot{q} = F_{tot}(q) \tag{19}$$

Technically, all these methods can be used both for *online* and *offline motion planning*. Using Eq. 17 we directly obtain control inputs for the robot: generated paths are smoother thanks to the "filtering" effect of the robot dynamics. If we would use Eq. 18 we should solve an inverse dynamic problem, substituting $\ddot{q}$ in the robot dynamic order in order to get forces. Eq. 19 could provide reference inputs to low-level controllers. In absence of local minima only Eq. 19 guarantees *asymptotic stability* of q_{goal} being, among other things, the fastest method. For the latter consideration, in

this dissertation the third option has been chosen. If position formation PIDs in Sect. V are modified in order to accept *velocities as inputs* and the APF algorithm generates velocities as outputs, the follower aircraft can be controlled.

VII. Formation control using a two-dimensional Artificial Potential Field Algorithm

A. Code architecture

Once the APF method has been defined, the cooperative formation algorithm should be created. Our algorithm will be essentially made up of five functions:

- *Grid Generator*
- *Objects (relative) position calculator*
- *Potential Generator*
- *Potential Solver*
- *Formation control block*

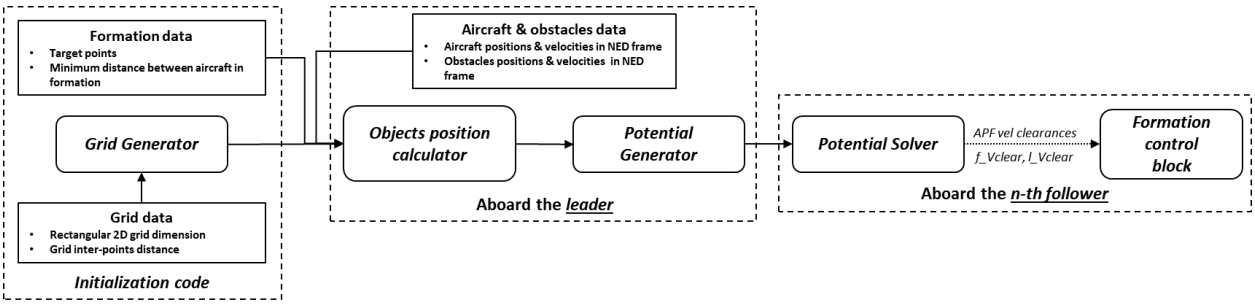


Fig. 5 Formation control algorithm architecture

In order to calculate the potential function, as can be observed in Eq. 14-15, the space should be *discretized* transforming it in a two- or three-dimensional tensor. The *Grid Generator* inside the *initialization code* fulfils this task, creating a series of points: in so doing, a potential value corresponds to each element of the grid. It should be noticed that the grid itself does not change during the simulation, in contrast to the potential.

The *leader aircraft*, considered as the most powerful element of the formation (from a computational point of view), includes two important functions: *Objects (relative) position calculator* and *Potential Generator*. As the grid itself is built around the leader, we need *relative* distances between a point in the grid $\boldsymbol{x}$, the target point $\boldsymbol{x}_{goal}$ and obstacles $\boldsymbol{x}_{obs}$. As will be described later, every potential is computed in a *leader-centred reference frame*. In fact, the leader is considered as the most advanced aircraft in the formation, to have the best computational capacity. The *Objects (relative) position calculator* block is responsible to obtain these values, subtracting the $X_{L,NED}$ and $Y_{L,NED}$ positions of the leader to the follower ones and rotating the results using ψ_L. The *Potential Generator* block calculates the total potential in every point of the grid. After defining all the constants K_a, K_{r1} and K_{r2} for leader and follower, the repulsive radius r_0 and after preallocating matrices, potential values are computed for every point of the grid. Moreover, when dealing with repulsive terms, only points inside r_0 are selected. The attractive potential for the target point is calculated in a similar manner and the total potential for every follower is obtained adding all the terms.

Every *follower aircraft* integrates other two functions: the *Potential Solver* and the *Formation control block*. The first function has the task of calculating the steepest descent direction in the nearest point of the grid where the aircraft is placed. Suppose that, for every simulation step of the APF (i.e. being $f = 1\,Hz$, every second), we obtain the entire route. In order to calculate errors for formation PIDs, we should select a point from which we could extract two- or three-dimensional values (one for each direction). These values can be:

- *position* of one point P of the route
- *force* F of one point P of the route

Formation PIDs have to be created depending on the type of value we have chosen. In our case both solutions have been investigated but it has been decided to extract the *force* F so as to be coherent with Siciliano [25]. The algorithm only performs a single iteration. Moreover, the formation control algorithm works at a frequency $f = 1\,Hz$. Finally, the *Formation control block* receives the *steepest descent direction* and transforms it in valid commands for the autopilots.

B. The Simulink model - time varying 2D scenario

At this point, the time varying 2D case will be described. The aircraft model is the same used for the *PID formation control model* (Sect. V), with all its mobile surfaces and autopilot blocks. In this case, however:

- Relative distances and potentials in *leader-centred reference frame* are calculated by the leader. The *APF leader* block has been added.
- Potential tensors are transmitted to every follower and solved using the *gradient descent* algorithm. Only one step is executed and results are conveniently handled to be used by formation PIDs

Following the description in Sect. VII.A, first of all a grid is created (*Grid Generator*) in the initialization code. Relative distances are computed (*Objects (relative) position calculator* block) and, at this point, the potential should be calculated (*Potential Generator* block). Every aircraft, during the simulation, can change its velocity acting on the throttle. If an obstacle must be avoided at higher velocities then V_0, a faster platform response time is needed. In our model the inertia of the aircraft has been implicitly taken into consideration during all PID tuning and it mainly depends on plant matrices, which are constant. In order to facilitate obstacle avoidance, the potential generator function has been modified acting on the *repulsive radius*. Supposing a velocity interval of $[60 - 140]\ \frac{m}{s}$, the repulsive radius of every aircraft is calculated by a exponential function:

$$R_{rep} = R_{obs} + toll = R_{max} \left(\frac{R_{min}}{R_{max}} \right)^{\frac{V-140}{60-140}} \tag{20}$$

R_{max} and R_{min} should be chosen accurately in order not to interfere with the formation shape logic. If we choose a wrong value of R_{max}, for example, followers could have difficulty reaching their target points. In our code it has been decided to compute R_{max} depending on the minimum aircraft distance in the formation patten (called *minfiguredist*). $R_{min,L} = 25\,m$ and $R_{min,F} = 20\,m$ are constant values while R_{max} varies:

$$\begin{aligned} R_{max,L} &= \max\left(25, \min\left(50, minfiguredist\right)\right) \\ R_{max,F} &= \max\left(20, \min\left(40, minfiguredist\right)\right) \end{aligned} \tag{21}$$

This means that if $V > V_0$ the repulsive radius increases and every follower "perceives" obstacles earlier, having more time to avoid them.

In the *Potential Solver* block a modified version of the gradient descent algorithm (Algorithm 1) is included. First of all, the index of the current position of the follower $x_{current}$ is found. Using the Matlab function *gradient*, the absolute value of the force $F_{current}$ is calculated (it is limited to 10^{-6} in order to avoid NaN errors) and the *descent direction dL* is obtained. As we described in Sect. VI.B, the output of the APF depends on $F_{current}$ and consequently on *dL*. The main problem is that formation PIDs need vectors in *follower-centred frame*, so the descent direction should be rotated obtaining the direction d:

$$\boldsymbol{d} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}_{FOLLO} = \boldsymbol{R_{LF}} \cdot \boldsymbol{dL} = \begin{bmatrix} \cos\left(\psi_F - \psi_L\right) & \sin\left(\psi_F - \psi_L\right) \\ -\sin\left(\psi_F - \psi_L\right) & \cos\left(\psi_F - \psi_L\right) \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}_{LEAD} \tag{22}$$

At this point we should choose the type of command to send to the follower, which will characterize the input of the *Formation control block*. In Sect. VI.B we have described three *planning techniques* and we said that the best option for our case was Eq. 19:

$$\dot{q} = F\left(q\right) \tag{19}$$

The force F must be calculated. These velocities (along X and Y) lead the follower to fly along a specific direction which is the one calculated by APF in order to avoid obstacles and reach safely the target point:

$$\begin{aligned} f_Vclear &= min(5, 0.15 * abs(F(1))) * d(1) \\ l_Vclear &= min(6, 0.20 * abs(F(2))) * d(2) \end{aligned} \tag{23}$$

As can be observed from Eq. 23, f_Vclear and l_Vclear are obtained multiplying d by a term which depends on the corresponding F component. Using the Matlab *min* function we can insert a superior limit, avoiding excessive high speed.

f _Vclear and *l _Vclear*, that are velocities in *follower-centred reference frame*, can be used to calculate errors needed by the *Formation control block*. Leader and followers NED velocities, obtained from their plant blocks, are subtracted and rotated using the heading angle ψ_F:

$$\begin{bmatrix} \Delta V_{N,followerframe} \\ \Delta V_{E,followerframe} \end{bmatrix} = \boldsymbol{R}_{LF} \cdot \Delta \boldsymbol{V}_{NED} = \\ = \begin{bmatrix} cos(\psi_F) & sin(\psi_F) \\ -sin(\psi_F) & cos(\psi_F) \end{bmatrix} \begin{bmatrix} V_{N,F} - V_{N,L} \\ V_{E,F} - V_{E,L} \end{bmatrix} \tag{24}$$

Altitude error does not need to be rotated:

$$\Delta h = h_F - h_L \tag{25}$$

Formation errors are calculated:

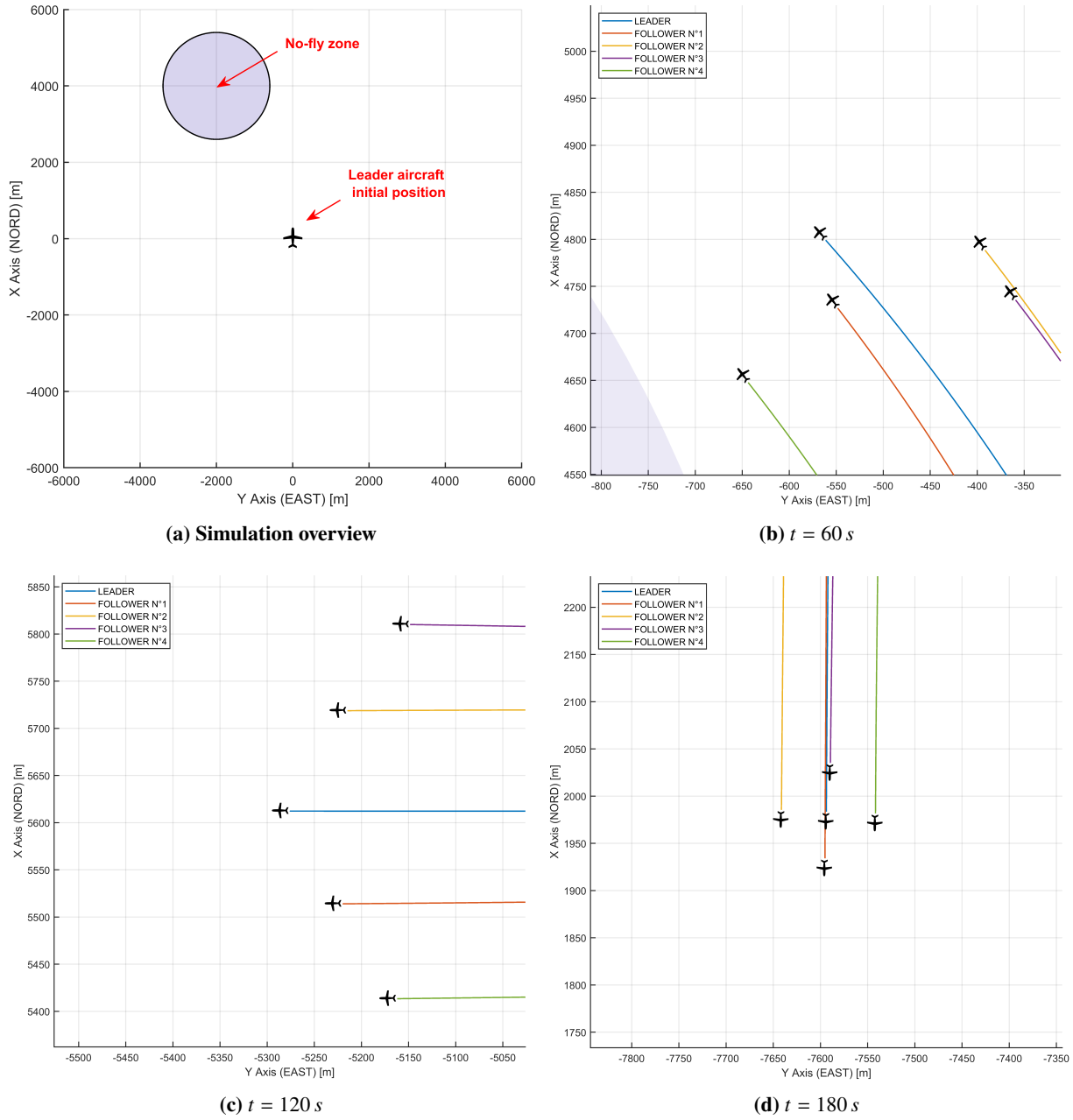
$$\begin{cases} \dot{f}_{formationerror} = \dot{f}_{APF} - \Delta V_{N,followerframe} \\ \dot{l}_{formationerror} = \dot{l}_{APF} - \Delta V_{E,followerframe} \\ h_{formationerror} = h_{APF} - \Delta h \end{cases} \tag{26}$$

where $h_{APF} = 0\,m$ so that every follower can flight at the same altitude of the leader. These errors can be used in formation PIDs. The *Lateral velocity formation PID* consists of a proportional, a derivative and an integral line. The output, $\Delta\varphi$, is added to the *Heading Autopilot* one. Thanks to a *tuning* procedure, gains have been calculated. The *Forward velocity formation PI* does not include a derivative line as it has been observed that it causes signal disturbances. The output of this block is directly sent to the *Throttle LQR* block.

C. Simulation

In Fig. 6a an overview of the scenario at $t = 0\,s$ is depicted. The leader aircraft position in NED reference frame is $[0, 0, H_0]$ ($H_0 = 3048\,m$) while followers are in *wall* formation. A *no-fly zone*, considered as a cylinder with infinite height and radius $R_{nfz} = 1400\,m$ has been centred in $C_{nfz} = [x_{nfz}, y_{nfz}] = [4000, -2000]\,m$. The *repulsive radius* of the *no-fly zone* has been calculated doubling R_{nfz}. The manoeuvre vector consists of three parts. First of all, every aircraft flights straight without any change in velocity. At $t = 40\,s$ a coordinated turn is performed toward $\psi = -90\,deg$ avoiding the *no-fly zone* and at $t = 120\,s$ a second turn is performed toward $\psi = -180\,deg$. At the same time the formation shape changes. Starting from the *wall* formation, at $t = 5\,s$ a *vic* geometry is ordered while at $t = 120\,s$ the selected shape is *box*.

In Fig. 6 results of the simulation can be observed. For the sake of simplicity, the followers are labelled with abbreviations F1, F2, F3, F4 (e.g. F1 is the first follower). If the *wall* formation at $t = 0\,s$ is the initial shape, the formation subsequently receives the order to generate a *vic* one. At $t = 20\,s$ F2 has successfully reached its position while F1, F3 and F4 repel themselves. The closest distance between F1 and F3 obtained during the manoeuvre strongly depends on attractive/repulsive constants, APF gains and aircraft inertia and could be reduced modifying these values. The *vic* pattern at $t = 40\,s$ is not complete yet but the leader begins its left turn. The *no-fly zone* influences the formation, as can be seen in Fig. 6b: every aircraft moves right in order to avoid the *no-fly zone*. At $t = 80\,s$ the first follower is on the right side of the leader while it should be on the left according to the *vic* formation. When the obstacle has been avoided and the heading direction is the desired one ($\psi = -90\,deg$ for the first turn), the correct *vic* shape is recovered (Fig. 6c) . At this point a second turn and a *box* shape are ordered and in Fig. 6d it can be seen how aircraft succeed in it, despite the short distance between them. This simulation demonstrates how this algorithm works in a complex scenario. Every follower tends to stay close to the leader and avoid obstacles with the possibility to change the formation shape in every moment. Varying few constants, the algorithm could be easily adapted to various type of aircraft because it does not depend directly to the plant geometry.



(a) Simulation overview

(b) $t = 60\,s$

(c) $t = 120\,s$

(d) $t = 180\,s$

Fig. 6 APF 2D - Simulation results

VIII. Formation control using a three-dimensional Artificial Potential Field Algorithm

A. Code adaptations

Having verified the operating principle of the two-dimensional APF-based cooperative formation control algorithm it has been decided to extend its functioning to a three-dimensional case. The plant model of the aircraft does not change in contrast to all the APF blocks. Before describing how the Simulink model has been adapted, we will briefly focus on the grid generation. If in the previous model the space was modelled as a *square*, in this case we deal with a *parallelepiped* filled with equidistant points whose number is higher than the 2D model one (with equal XY dimensions). For this reason, 3D grids will be smaller and coarser than two-dimensional ones.

As discussed in previous chapters, the grid is generated in order to be leader-centred and it should be rotated so that

it can be always aligned with leader body axes. In the 3D case we need to consider other angles, as *pitch* θ_L and *roll* φ_L. The leader aircraft, as in the 2D case, is responsible for calculating relative distances and potentials. In the 2D APF Simulink model altitude h was considered constant and obstacles were defined using initial positions and velocities in *north-east* plane; quite the opposite, in this case obstacles are defined using $h_{0,obs}$ and $\dot{h}_{0,obs}$ also. The *Objects (relative) position calculator* block has been modified in order to consider a constant vertical velocity. The potential also should be calculated taking altitude into consideration, and it has been done modifying the *Potential Generator* block. Regarding the follower aircraft, both *Potential Solver* and *Formation control* blocks has been adapted to the 3D case. An additional velocity command, *h_Vclear*, has been added to the algorithm to command the vertical velocity for each aircraft.

B. Simulation

In order to present how the model works, a three-dimensional simulation has been performed. The grid, as previously outlined, is generally coarser than the two-dimensional version: in this case a point-to-point distance of $4\,m$ has been selected. In Fig. 7 an overview of the scenario is depicted. A *moving obstacle* with physical radius $R_{obs} = 30\,m$ (and repulsive radius $R_{pot} = 60\,m$) has been inserted in $C_{mo} = [x_{mo}, y_{mo}] = [-200; 70]\,m$, having a velocity $V = V_{NORD} = 92.3111\,\frac{m}{s}$. The initial formation is of *wall* type and it is kept until $t = 25\,s$, when an *arrow* is ordered; finally, at $t = 70\,s$, a *line astern* shape is executed. As far as manoeuvres are concerned, every aircraft initially flights straight, without changing its altitude. At $t = 20\,s$ the leader starts a right turn with altitude change ($\dot{h} = 2\,\frac{m}{s}$), obtaining an *heading angle* $\psi_L = 60\,deg$. At a later time ($t = 90\,s$) a *flare* is performed.

The scenario has been simulated and it can be observed in Fig. 8. While all aircraft are initially in *wall* formation the moving obstacle, whose velocity is greater than the aircraft one, reaches them from behind. F3 and F4 do not perceive the object while F1 and F2, as can be seen in Fig. 8a, avoid it. After avoiding the obstacle, a right turn is ordered together with the *arrow* shape. At $t = 40\,s$ every aircraft reaches its *desired point* during the turn. The increase in altitude does not affect the formation shape, which is almost completed in Fig. 8b. At $t = 70\,s$ the *arrow* shape is broken and gradually changed to a *line astern* type. The effect of the APF along Z is extremely important here. The position of the first follower (in *leader-centred reference frame*) in the *arrow* is $TP_{F1,arrow} = [75; 0; 0]\,m$ while the subsequent *line astern* desired point is $TP_{F1,line\,astern} = [-30; 0; -20]\,m$. Using a three-dimensional APF altitude can be changed: F1 avoids the leader reducing it. At $t = 90\,s$ the *flare* manoeuvre is activated, and its effect is evident in Fig. 8c. In this case the *line astern* formation is successfully completed and kept (Fig. 8d) during the manoeuvre.

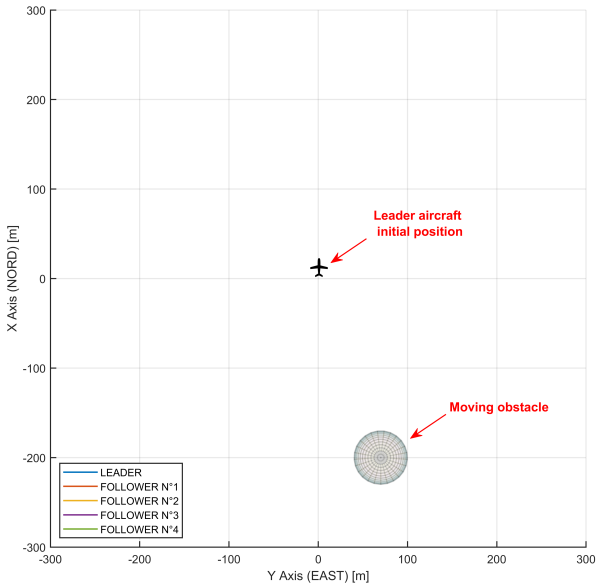
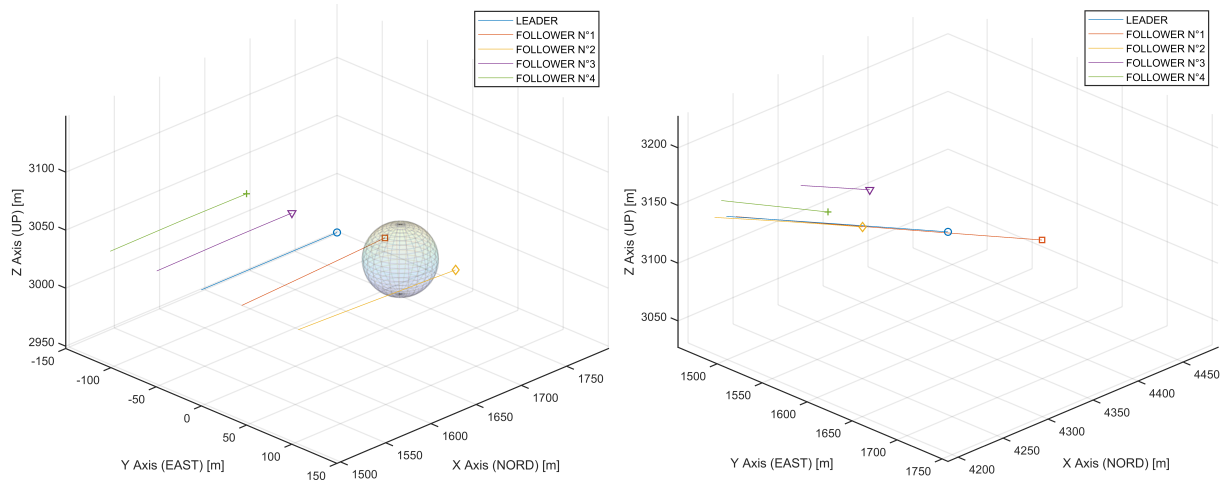
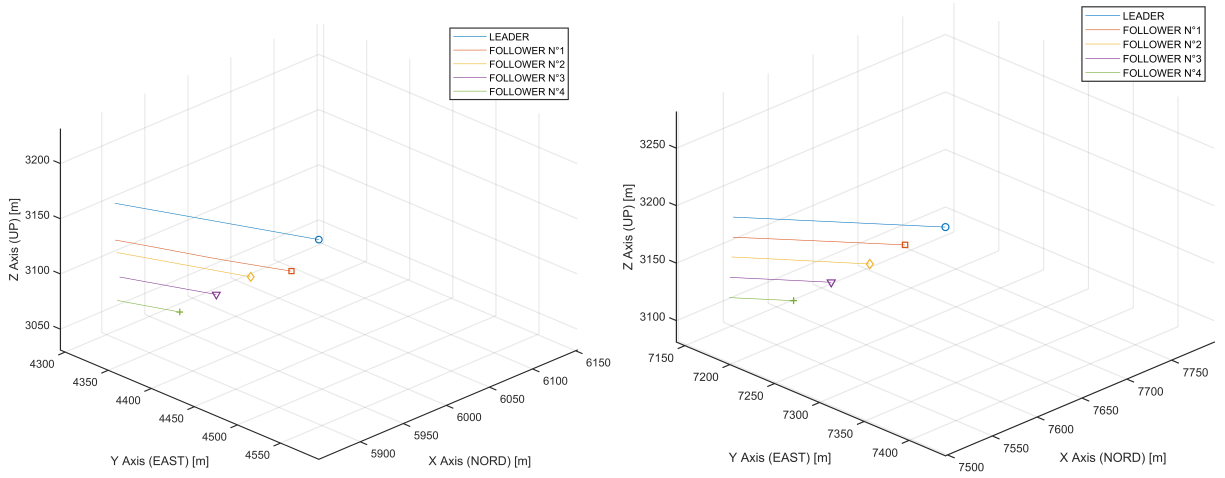


Fig. 7 Simulation overview (APF 3D)



(a) $t = 20\,s$ **(b)** $t = 60\,s$



(c) $t = 100\,s$ **(d)** $t = 140\,s$

Fig. 8 APF 3D - Simulation results

IX. Conclusions

This work deals with the design and modelling of a formation control algorithm of multiple UAVs based on the Artificial Potential Fields (APF) method. The proposed approach has been investigated in 2D and 3D space along with the chance to simulate no-fly zones and moving obstacles. In this scenario, some potential functions are proposed to control the formation flight. The algorithm demonstrated to have promising capabilities to deal with operative scenario where a manned aircraft is the leader of a formation made up of unmanned aircraft. The time-varying scenario (due to no-fly zones or intruders) is quite manageable for 2D problems, whereas it becomes time consuming in 3D space. In fact, the main drawback of the proposed solution is the limit for real time applications that could be further investigated considering other type of space discretisation. Moreover, the proposed potential functions can be optimised to improve the overall formation control performance.

References

[1] NASA, "Past Projects: Autonomous Formation Flight (AFF)," , 2017. Https://www.nasa.gov/centers/dryden/history/ pastprojects/AFF/index.html - checked April 16, 2023.

[2] Blake, Bieniawski, "Surfing aircraft vortices for energy," *Journal of Defence Modeling and Simulation: Applications, Methodology, Technology*, Vol. 12, 2015, pp. 31–39.

[3] Schkolnik, Cobleigh, "Autonomous Formation Flight: a primary goal is to reduce fuel consumption during cruise by 10 percent," *NASA Tech Briefs*, 2004, pp. 20–21.

[4] Jenkinson, Caves, Rhodes, "Automatic formation flight - A preliminary investigation into the application to civil operations," *AIAA*, 2012.

[5] NASA, "AARD - Autonomous Airborne Refueling Demonstration," , 2007. Https://ntrs.nasa.gov/api/citations/20070025036/ downloads/20070025036.pdf - checked April 16, 2023.

[6] Tsukerman et al., "Optimal Rendezvous Guidance Laws with Application to Civil Autonomous Aerial Refueling," *Journal of Guidance,Control and Dynamics*, Vol. 41, 2018.

[7] Perret, B., "Moving Fast," *AviationWeek.com*, 2022.

[8] Liu, Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica*, Vol. 36, 2018, pp. 1019 –1047.

[9] Wang, "Navigation strategies for multiple autonomous mobile robots moving in formation," *Journal of Robotic Systems*, Vol. 8, 1991.

[10] Gu, Campa, Napolitano et al., "Design and flight-testing evaluation of formation control laws," *IEEE Transactions on Control Systems Technology*, Vol. 14, 2006.

[11] Desai et al., "Controlling formations of multiple mobile robots," *IEEE International Conference on Robotics and Automation*, 1998.

[12] Dang, Horn, "Formation Control of Leader-Following UAVs to Track a Moving Target in a Dynamic Environment," *Journal of Automation and Control Engineering*, Vol. 3, 2014.

[13] Lewis, Tan, "High Precision Formation Control of Mobile Robots Using Virtual Structures," *Autonomous Robots*, Vol. 4, 1997, pp. 387–403.

[14] Balch, Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, Vol. 55, 1998, pp. 926–939.

[15] K.D.Do, J.Pan, "Nonlinear formation control of unicycle-type mobile robots," *Robot. Auton. Syst.*, Vol. 55, 2007, pp. 191–204.

[16] Liu, *Robot Systems for Rail Transit Applications*, Elsevier, 2020.

[17] Garrido, Moreno, Lima, "Robot formation motion planning using fast marching," *Robotics and Autonomous Systems*, Vol. 59, 2011, pp. 675–683.

[18] Schouwenaars, How, Feron, "Receding Horizon Path Planning with Implicit Safety Guarantees," *Proceedings of the American Control Conference*, Vol. 6, 2004, p. 5576–5581.

[19] Dakulović, Petrović, "Two-way D star algorithm for path planning and replanning," *Robotics and Autonomous Systems*, Vol. 59, 2011, p. 329–342.

[20] Qureshi, Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, Vol. 68, 2015, pp. 1–11.

[21] Nelson, R., *Flight stability and automatic control*, WCB/McGraw Hill, 1998.

[22] Tedrake, R., *Underactuated Robotics*, 2023. URL `https://underactuated.csail.mit.edu`.

[23] Campa, Napolitano et al., "Development of Formation Flight Control Algorithms Using 3 YF-22 Flying Models," 2007.

[24] Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, Vol. 5, 1986, pp. 396–404.

[25] Siciliano, Sciavicco et al., *Robotics - Modelling, planning and control*, Springer, 2009.

[26] Punzo, "Verifiable Swarm Engineering with Limited Communication," 2013.

[27] Choset et al., *Principles of Robot Motion - Theory, Algorithms and Implementation*, MIT Press, 2005.