A software-in-the-loop implementation of adaptive formation control for fixed-wing UAVs

Yang, Jun; Wang, Ximan; Baldi, Simone; Singh, Satish; Fari, Stefano

DOI
10.1109/JAS.2019.1911702

Publication date
2019

Document Version
Final published version

Published in
IEEE/CAA Journal of Automatica Sinica

A Software-in-the-Loop Implementation of Adaptive Formation Control for Fixed-Wing UAVs

Jun Yang, Ximan Wang, Simone Baldi, Satish Singh, and Stefano Farì

Abstract— **This paper discusses the design and software-in-the-loop implementation of adaptive formation controllers for fixed-wing unmanned aerial vehicles (UAVs) with parametric uncertainty in their structure, namely uncertain mass and inertia. In fact, when aiming at autonomous flight, such parameters cannot assumed to be known as they might vary during the mission (e.g. depending on the payload). Modeling and autopilot design for such autonomous fixed-wing UAVs are presented. The modeling is implemented in Matlab, while the autopilot is based on ArduPilot, a popular open-source autopilot suite. Specifically, the ArduPilot functionalities are emulated in Matlab according to the Ardupilot documentation and code, which allows us to perform software-in-the-loop simulations of teams of UAVs embedded with actual autopilot protocols. An overview of realtime path planning, trajectory tracking and formation control resulting from the proposed platform is given. The software-inthe-loop simulations show the capability of achieving different UAV formations while handling uncertain mass and inertia.**

Index Terms—**ArduPilot, adaptive formation control, Fixed-wing UAVs, software-in-the-loop simulations.**

I. Introduction

UNMANNED aerial vehicles (UAVs) are generating the curiosity of several scientific communities. Among the various types of UAVs, fixed-wing UAVs have been studied in different contexts from military to commercial, due to their energy efficient performance while carrying payloads [1], [2]. The holy grail of such researches is to have formations of UAVs that are able to complete missions autonomously with little supervision from the human operator [3]. As such, it is necessary to equip UAVs with a smart flight control unit. Path following is one of the most crucial tasks for implementation in flight control units: many mature control theories and algorithms have been proposed for path following. In [4], state-of-the-art path following algorithms are summarized and compared with each other using two metrics: control effort and cross-track error. Five algorithms are evaluated, namely carrot-chasing, nonlinear guidance, vector-field (VF), linear quadratic regulation and pure pursuit with line-of-sight (cf. [5]–[16] for more details on such algorithms and on variants of such algorithms). Monte Carlo simulations in [4] show that the VF path following, a technique developed in [17], is more accurate than the other methods, while requiring more parameters to be designed. The basic concept of VF path following is to construct a vector field around the desired path, resulting in course commands to the vehicle. Path following laws are typically derived from Lyapunov stability analysis which guarantees stable convergence to the desired path. Despite the advances in the field, several challenges remain in path following. For example, the simulations in [4] and in the aforementioned works highlight three crucial points:

a) The actual performance of path-following methods considerably depends on the fidelity of the UAV model used for design. When aiming at autonomy, parametric uncertainties will inevitably appear in the UAV structure (uncertain mass and inertia might vary during the mission). Path-following algorithms that cannot adapt to such changes will exhibit poor performance.

b) The actual path-following performance depends not only on the commanded UAV course angle. At a lower level, a complex suite of algorithms commonly referred to as *autopilot*, must be in charge of regulating roll, pitch and altitude (rudder/wing/aileron actuators) according to the course commanded by the path-following algorithm.

c) Simulations performed on single UAVs or teams of UAVs to test path-following protocols usually do not include the autopilot layer [4]; this testing is to a large extent open.

Given these challenges, this work is driven by the following research questions: how to cope with parametric uncertainties in the UAV? How to account for the autopilot low-level control when testing path-following algorithms? How to scale the path-following problem to teams of UAVs? While some of the authors studied in [18]–[24] adaptive formation control algorithms for various systems with uncertain dynamics, the corresponding problem for UAVs is much more challenging due to the complex UAV control architecture as sketched in Fig. 1. This architecture relies on multiple layers: the autopilot contains the low-level control algorithms that are able to

Manuscript received April 16, 2019; revised May 19, 2019; accepted July 21, 2019. This work was supported by the Fundamental Research Funds for the Central Universities (4007019109) (RECON-STRUCT), the Special Guiding Funds for Double First-class (4007019201), and the Joint TU Delft - CSSC Project 'Multi-agent Coordination with Networked Constraints' (MULTI-COORD). The first two authors equally contributed to this work. Recommended by Associate Editor Wei He. *(Corresponding author: Simone Baldi.)*

Citation: J. Yang, X. Wang, S. Baldi, S. Singh, and S. Farì, "A software-in-the-loop implementation of adaptive formation control for fixed-wing UAVs," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 5, pp. 1230–1239, Sept. 2019.

J. Yang is with Systems Engineering Research Institute, China State Shipbuilding Corporation, Beijing 100094, China (e-mail: yangjun@sohu.com).

X. Wang and S. Singh are with Delft Center for Systems and Control (DCSC), TU Delft, 2628CD Delft, The Netherlands (e-mail: x.wang-15@tudelft.nl; satish1989221@gmail.com).

S. Baldi is with School of Mathematics, Southeast University, 211189 Nanjing, China, and also with DCSC, TU Delft, 2628CD Delft, The Netherlands.(e-mail: s.baldi@tudelft.nl).

S. Farì is with German Aerospace Center (DLR), Institute of Space Systems, D-28359 Bremen, Germany, and was with Politecnico di Milano, Italy and also with DCSC, TU Delft, The Netherlands (e-mail: stefano.fari@mail.polimi.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2019.1911702

maintain roll and pitch angles, airspeed, altitude, and roll. UAV states (or estimated states) and cross-track errors is the crucial information to be used by the higher levels. The path following is meant to maintain the vehicle on the desired path by providing the course heading; the path manager supervises the navigation of the UAV with a finite-state machine which converts a sequence of way-points into a sequence of path primitives that the path following can track.

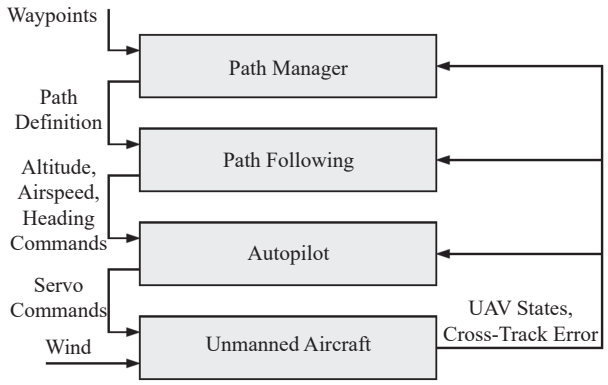

Fig. 1. General layout for UAV control with autopilot. In this work we show how, by reverse engineering the Ardupilot code, one can perform software-in-the-loop simulations with the actual autopilot protocols of the UAV.

Having highlighted how modelling and autopilot design are crucial steps towards the autonomous control of fixed-wing UAVs, this papers exactly addresses such issues for a team of autonomous fixed-wing UAVs. The modelling is implemented in Matlab, while the autopilot algorithms are taken by ArduPilot, a popular open-source autopilot suite: specifically, the ArduPilot functionalities are replicated in Matlab following the Ardupilot documentation and the Ardupilot code itself (reverse engineering), a feature that allows us to perform software-in-the-loop simulations with the actual autopilot protocols. Such software-in-the-loop simulations show the capability of handling parametric uncertainty in the UAV structure, (i.e. handling uncertain mass and inertia) for a team of UAVs.

The rest of the paper is organized as follows: Section II gives some details on the Matlab UAV simulation platform. Section III describes some aspects related to the autopilot, while Section IV discusses hardware and software integration of the various components. In Section V an algorithm for adaptive vector field path following is given, followed by an adaptive formation control method in Section VI, with simulations tests. Section VII prospects future research directions.

II. Modelling

In line with Fig. 1, the basic level of a reliable fixed-wing UAV simulator must contain the UAV dynamics and the dynamics of the environment (wind). These are briefly sketched below, in conformity with the standard literature [25], [26]).

A. Equations of Motion

Using the variables in Table I, the motion of a fixed-wing

TABLE I
State Variables for Equations of Motion

State	Description
ϕ	Euler angle for Roll
θ	Euler angle for Pitch
ψ	Euler angle for Yaw
$\bar{u}$	Angular velocity along x-axis in body frame
$\bar{v}$	Angular velocity along y-axis in body frame
$\bar{w}$	Angular velocity along z-axis in body frame
$\bar{p}$	Roll rate along x-axis in body frame
$\bar{q}$	Pitch rate along y-axis in body frame
$\bar{r}$	Yaw rate along z-axis in body frame

UAV can be written in the Euler-Lagrange (EL) form as [25]:

$$\underbrace{\begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x & 0 & -I_{xz} \\ 0 & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & -I_{xz} & 0 & I_z \end{bmatrix}}_{D(q)} \underbrace{\begin{bmatrix} \dot{\bar{u}} \\ \dot{\bar{v}} \\ \dot{\bar{w}} \\ \dot{\bar{p}} \\ \dot{\bar{q}} \\ \dot{\bar{r}} \end{bmatrix}}_{\ddot{q}} +$$

$$\underbrace{\begin{bmatrix} 0 & -m\bar{r} & m\bar{q} & 0 & 0 & 0 \\ m\bar{r} & 0 & -m\bar{p} & 0 & 0 & 0 \\ -m\bar{q} & m\bar{p} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_z\bar{r} - I_{xz}\bar{p} & -I_y\bar{q} \\ 0 & 0 & 0 & I_{xz}\bar{p} - I_z\bar{r} & 0 & I_x\bar{p} - I_{xz}\bar{r} \\ 0 & 0 & 0 & I_y\bar{q} & I_{xz}\bar{r} - I_x\bar{p} & 0 \end{bmatrix}}_{C(q,\dot{q})}$$

$$\underbrace{\begin{bmatrix} \bar{u} \\ \bar{v} \\ \bar{w} \\ \bar{p} \\ \bar{q} \\ \bar{r} \end{bmatrix}}_{\dot{q}} + \underbrace{\begin{bmatrix} mg\sin(\theta) \\ -mg\sin(\phi)\cos(\theta) \\ -mg\cos(\phi)\cos(\theta) \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{g(q)} = \begin{bmatrix} F_{\tau_1} \\ F_{\tau_2} \\ F_{\tau_3} \\ M_{\tau_1} \\ M_{\tau_2} \\ M_{\tau_3} \end{bmatrix}, \quad \tau = \begin{bmatrix} F_{\tau_1} \\ F_{\tau_2} \\ F_{\tau_3} \\ M_{\tau_1} \\ M_{\tau_2} \\ M_{\tau_3} \end{bmatrix} \tag{1}$$

where m is the mass of the UAV, F_{τ_1}, F_{τ_2}, and F_{τ_3} are the forces acting in x, y, z coordinate axes; I is the inertia tensor and M_{τ_1}, M_{τ_2} and M_{τ_3}, are the moments acting in x, y, z axes. It is taken into account that the fixed-wing UAV is symmetric with respect to x and z axes and inertia in the planes xy and yz is negligible. As wind may represent 20%-50% of the airframe airspeed, wind is included in the simulation, by modelling it as the composition of a constant part and a dynamic part (known in literature as Dryden model [27]). Along the lines of [26], in order to properly describe the influence of the wind, one needs to define the ground speed, i.e. the UAV velocity relative to the inertial frame. Such ground speed is commonly denoted with $\mathbf{V}_g$, and it is a crucial variable when deriving the path-following laws [4].

B. Matlab-Based Simulator

The fixed-wing UAV and wind dynamics have been implemented in the Matlab-Simulink environment by means

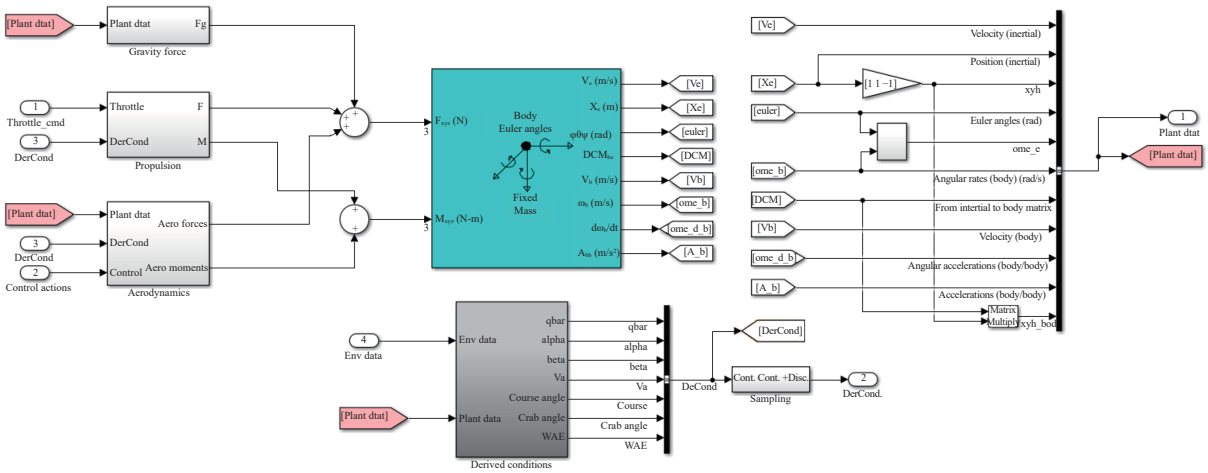

Fig. 2. Matlab Simulink model for UAV dynamics. The model comprises the forces and moments on the UAV, as well as the airspeed, angle of attack, side-slip angle and course angle after the effect of the wind.

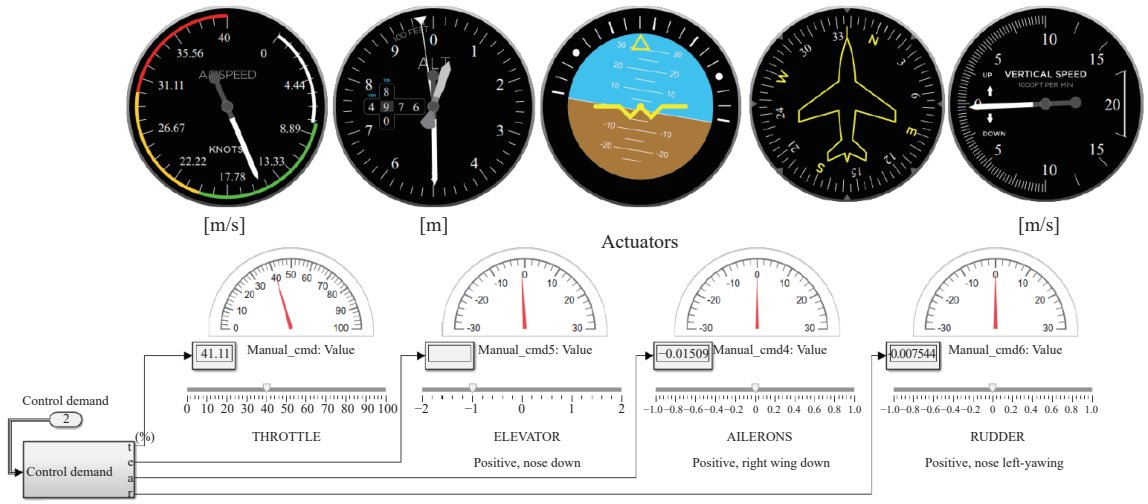

Fig. 3. Simulink visual interface.

of the Aerospace blockset [28]. Some screen-shots from the simulator can be seen in Fig. 2 and Fig. 3. In Fig. 2, the forces and moments contributions are shown on the left. On the right, the block 'Derived Conditions' contains the implementation of the wind dynamics, i.e. the computation of airspeed, angle of attack, side-slip angle, course angle, and other useful quantities affected by the wind. A visual interface, shown in Fig. 3, contains in-flight instruments embedded in the simulator, to help analyzing the flight status and reveal potential errors. More details on the simulator can be found in [29].

III. ARDUPILOT AUTOPILOT

Recalling that the final goal is to provide a realist UAV simulation platform, it is essential that the Matlab simulator can replicate the low-level control structure of the UAV (i.e. the autopilot layer). The code of ArduPilot, a professional autopilot software suite, is open-source and it thus can accessed and replicated in any other simulation platform. One of the main feature of ArduPilot is to let the user operate under different flight modes, which are:

Manual: The controller is not active, the pilot closes the loop. The radio controller stick commands of aileron, elevator, rudder and thrust are delivered to the control actuators as they are.

Fly-by-Wire A (FBWA): Control of roll and pitch angles is enabled, whose reference is given by the user with the radio controller stick commands.

Fly-by-Wire B (FBWB): In addition to roll and pitch angles, altitude and airspeed control is enabled, taking as reference the airspeed and rate of climb given by the user with the radio commands.

Autotune: Same as FBWA mode, but meanwhile the aircraft response is used to tune online the pitch and roll controllers.

Auto: The guidance logic is also enabled. The UAV will follow a set of GPS waypoints set by the user.

ArduPilot is written in C++, with many supporting utilities written in Python. In order to promote the integration of Ardupilot along with the aforementioned Matlab-Based UAV model, the ArduPilot functionalities are replicated in Matlab after studying the Ardupilot documentation [30] and the Ardupilot code itself [31]. This step of reverse engineering

allows us to perform software-in-the-loop simulations with the actual autopilot protocols of the UAV. A flowchart illustrating the structure of the ArduPilot is provided in Fig. 4.

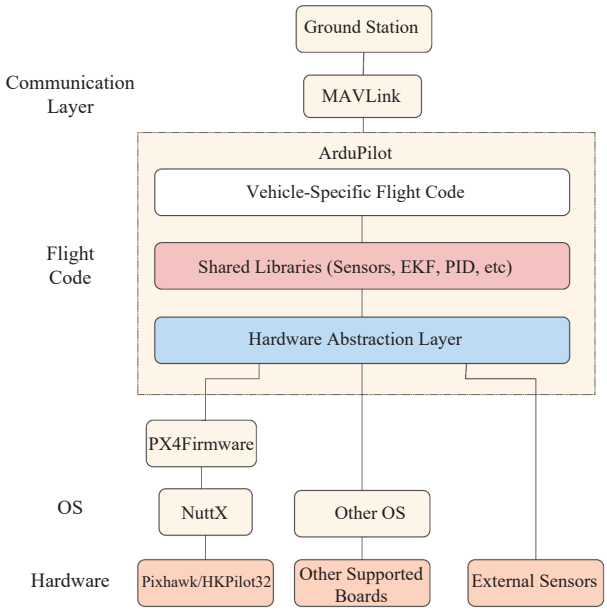

Fig. 4. Flowchart structure for the Ardupilot autopilot. Such a structure has been replicated in Matlab following the Ardupilot documentation.

Because the purpose of the autopilot layer is to provide low-level controllers to govern the various UAV states, let us illustrate the main ideas behind lateral and longitudinal UAV control. For most flight maneuvers of interest, autopilots are designed with the assumption of decoupled and linear lateral and longitudinal dynamics [26]. In this way, the autopilot design significantly simplifies. The decoupled linearized dynamics of the UAV are of first and second order

$$\text{roll} \quad \phi(s) = \frac{a_{\phi_2}}{s(s+a_{\phi_1})}\left(\delta_a(s) + \frac{1}{a_{\phi_2}} d_{\phi_2}(s)\right)$$

$$\text{pitch} \quad \theta(s) = \frac{a_{\theta_3}}{s^2 + a_{\theta_2} s + a_{\theta_1}}\left(\delta_e(s) + \frac{1}{a_{\theta_3}} d_{\theta_2}(s)\right)$$

$$\text{course} \quad \chi(s) = \frac{g}{V_g s}\left(\phi(s) + d_\chi(s)\right)$$

where the terms in d are disturbances coming from the coupled dynamics, and the definition for all variables can be found in [29]. Such first or second order loops allow an effective use of Proportional-Integral-Derivative (PID) control.

Let us focus only on the lateral dynamics, most relevant to path following: the roll controller structure is depicted in Fig. 5. It consists of two nested loops: the inner one controls the roll rate $\bar{p}$; the outer the roll angle ϕ; $C_{\phi_2}(z)$ is a PID controller; $\tilde{K}_{P_\phi}$ is a feed-forward gain; at the outer loop there is a proportional controller with gain Ω_ϕ. Similar reasoning applies to the pitch control scheme as shown in Fig. 6. The ArduPilot documentation provides descriptions on the structure of such loops and on the tuning of the PID controllers [32], which can then be perfectly replicated in

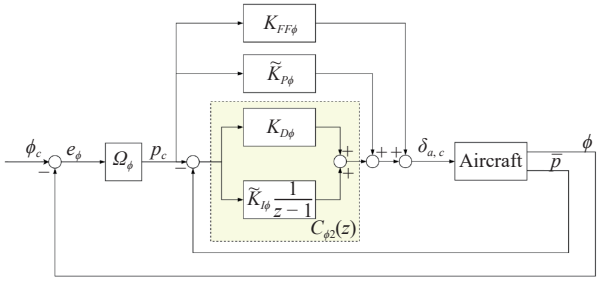

Fig. 5. Roll control scheme of the UAV. The variables ϕ, ϕ_c are the roll and commanded roll angle, while the variables $\bar{p}$, p_c are the roll rate and commanded roll rate. The commanded aileron is $\delta_{a,c}$.

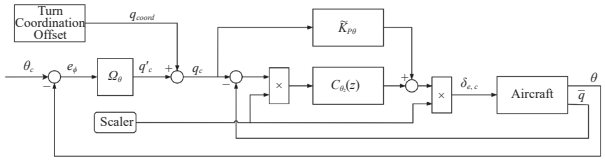

Fig. 6. Pitch control scheme of the UAV. The variables θ, θ_c are the pitch and commanded pitch angle, while the variables $\bar{q}$, q_c are the pitch rate and commanded pitch rate. The commanded elevator is $\delta_{e,c}$.

Matlab and eventually validated on a real fixed-wing UAV. Validation of both the roll and the pitch control loops has been performed on a HobbyKing Bixler UAV (cf. the detailed validation procedure in [29]), showing that the simulated fixed-wing UAV behaves very closely to the actual fixed-wing UAV.

IV. Hardware and Software Integration

This section presents the basic steps necessary for integration of hardware and software on an actual fixed-wing UAV with ArduPilot.

A. Flight Control Unit

ArduPilot can run on many different micro-controllers and platforms [33]. The HobbyKing HKPilot32 was chosen (see Fig. 7). It is a Pixhawk clone, an open-hardware flight controller specifically meant for UAV applications [34]. It has two redundant inertial measurement units (IMUs) which integrate a 3-axis accelerometer, a 3-axis gyroscope, and a magnetometer. The measurements from these devices are used by the state estimation protocols of ArduPilot to get the states

Fig. 7. The HobbyKing HKPilot32 micro-controller.

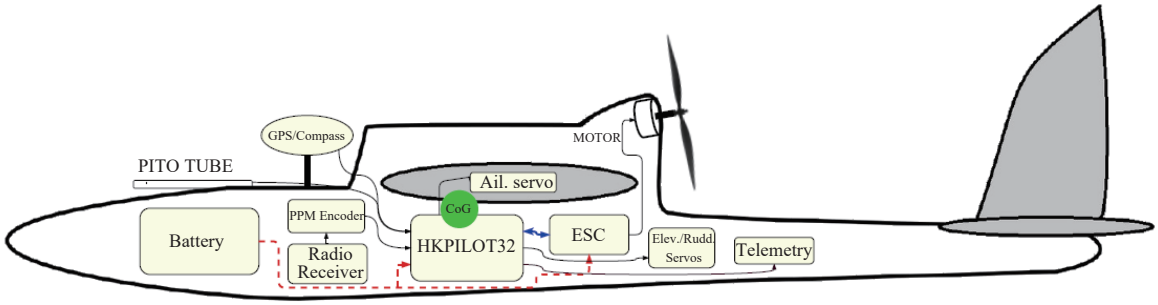

Fig. 8. Cross-section showing CoG and the distribution of electronics inside the UAV.

of the UAV. In fact, each accelerometer can output three acceleration measurements, one per axes, while the gyroscopes can measure the body angular rates on the three orthogonal axes. In HKPilot32 there is also a barometer for indirect altitude measurement. A real-time Operating System (OS) runs on HKPilot32, called NuttX: the OS is in charge of separating the program functions into self-contained tasks and implements an on-demand scheduling of their execution. The main benefit is that some tasks can be executed in parallel.

B. Integration

Integration of all the electronics submodules inside the airframe is shown in Fig. 8. As the HKPilot32 micro-controller contains the two IMUs necessary for the estimation of the plane attitude, it is advised to place it as close as possible to the center of gravity. It is also advised to place some foam dampers between the micro-controller and the fixing surface, at the corners. These dampers are required to: reduce sensor errors due to mechanical environment solicitations; protect sensors as they can be damaged by shocks or vibrations; contain parasitic IMUs movements. In fact, accelerometers are very sensitive to vibrations: in the presence of excessive vibrations, the state estimates can lead to very bad performance, thus preventing accurate positioning.

V. Vector-Field Path Following

As standard in literature, straight-line and orbit path are considered for path following [17]. VF strategies work under the assumption of first-order course χ dynamics

$$\dot{\chi} = \alpha_\chi(\chi_c - \chi), \tag{2}$$

with χ the course angle, χ_c the commanded course angle and α_χ the time constant. The main variables behind the VF path following are collected in Table II.

A. Straight Line Following

The vector field which describes the reference course to drive the UAV on the path is

$$\chi_d(e_{py}) = \chi_q - \chi_\infty \frac{2}{\pi}\tan^{-1}(k_{sl}e_{py}) \tag{3}$$

where e_{py} is the cross-track error, χ_q is the angle between the reference line and the north, χ_∞ is a parameter in $(0, \frac{\pi}{2}]$ which is the course reference when the error is large, and k_{sl} a tuning parameter governing the vector field smoothness. In [17] it is shown that the control law which is able to let $\chi \to \chi_d$ and $e_{py} \to 0$ as $t \to \infty$ is

$$\chi_c = \chi - \chi_\infty \frac{2}{\pi}\frac{\beta_s V_g}{\alpha_\chi}\sin(\chi - \chi_q) - \frac{\kappa_{sl}}{\alpha_\chi}\text{sat}\left(\frac{\tilde{\chi}}{\varepsilon_{sl}}\right) \tag{4}$$

where $\tilde{\chi} = \chi - \chi_d$, $\beta_s = k_{sl}/(1 + (k_{sl}e_{py})^2)$, $V_g = \|\mathbf{V}_g\|$. The parameters κ_{sl} and ε_{sl} govern the control aggressiveness and counteract possible chattering in the control action, and $\text{sat}(x) = x$, if $|x| < 1$ or $\text{sign}(x)$ otherwise.

TABLE II
Variables for Vector-Field Path Following

Variable	Description
χ	Course angle
χ_c	Commanded course angle
χ_d	Reference course angle (vector field)
χ_∞	Reference course at infinity
λ	=1 clockwise, =−1 counter-clockwise orbit
χ_q	Angle between reference line and the north
γ	Angle between UAV-center line and the north
$\tilde{\chi}$	Path-Following error (line)
$\tilde{s}$	Path-Following error (orbit)
k_{sl}, k_o	Vector field smoothness parameter
κ_{sl}, κ_o	Control authority parameter
$\varepsilon_{sl}, \varepsilon_o$	Anti-chattering parameter
V_g	Magnitude of ground speed

B. Orbit Path Following

The desired course vector field which drives the aircraft to loiter on an orbit path is

$$\chi_d(\tilde{s}) = \gamma + \lambda\left(\frac{\pi}{2} + \tan^{-1}(k_o\tilde{s})\right) \tag{5}$$

where is $\tilde{s} = s - R$, s is the distance of the UAV from the orbit center, R the orbit radius and γ is the angle between the north and the UAV position with respect to the orbit center. The parameter λ is 1 for clockwise orbit path and -1 for counter-clockwise orbit path. In [17] it is shown that the control law which is able to let $\chi \to \chi_d$ and $\tilde{s} \to 0$ as $t \to \infty$ is

$$\chi_c = \chi + \frac{V_g}{\alpha_\chi s}\sin(\chi - \gamma) + \beta_o\frac{\lambda V_g}{\alpha_\chi}\cos(\chi - \gamma) - \frac{\kappa_o}{\alpha_\chi}\text{sat}\left(\frac{\tilde{\chi}}{\varepsilon_o}\right) \tag{6}$$

where $\beta_o = k_o/(1 + (k_o\tilde{s})^2)$, and the parameters k_o, κ_o, ε_o are defined similarly to the straight-line case.

VI. Adaptive Formation Algorithm

In this section, a network formation of UAVs is considered, each one with dynamics:

$$D_i(q_i)\ddot{q}_i + C_i(q_i,\dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i, i = \{1,...,N\} \tag{7}$$

where the dynamics are in the EL form (1) as in Section II-A.

A. Preliminaries on Communication Graphs

The UAVs are linked to each other via a *communication graph* that describes the allowed information flow (cf. the example in Fig. 9). In a communication graph, a special role is played by the *pinning* node, which is a UAV (typically indicated as system 0) and it does not receive information from any other UAVs in the network. The communication graph describing the allowed information flow between all the systems, pinner excluded, is completely defined by the pair $\mathcal{G} = (\mathcal{V},\mathcal{E})$, where $\mathcal{V} = \{1,...,N\}$ is a finite non empty set of nodes, and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ is a set of pairs of nodes, called edges. To include the presence of the pinner in the network we define $\bar{\mathcal{G}} = \{\mathcal{V},\mathcal{E},\mathcal{T}\}$, where $\mathcal{T} \subseteq \mathcal{V}$ is the set of those nodes, called *target nodes*, which receive information from the pinner. Let us introduce the *Adjacency matrix* $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N\times N}$ of a directed communication graph, which is defined as $a_{ii} = 0$ and $a_{ij} = 1$, if $(i,j) \in \mathcal{E}$, where $i \neq j$. In addition, we define a vector, the *target vector* $\mathcal{M} = [a_{j0}] \in \mathbb{R}^N$, to describe the directed communication of the pinner with the target nodes. Specially, the target matrix is defined as $a_{jo} = 1$ if $j \in \mathcal{T}$ and $a_{jo} = 0$ otherwise.

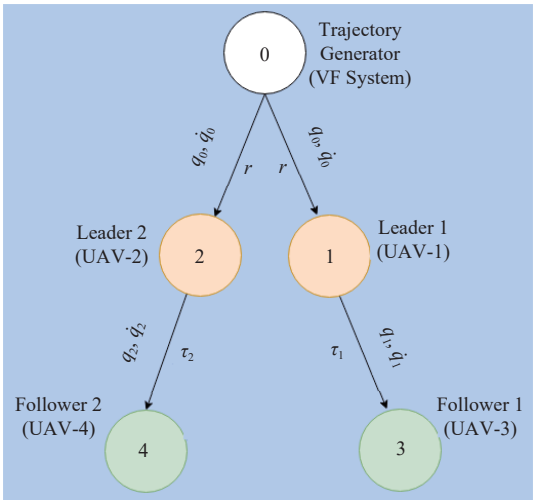

Fig. 9. Communication graph with V formation.

B. Formation Control Law

The main variables behind the formation control law are collected in Table III, whose explanation is sketched hereafter.

Given a hierarchical network $\bar{\mathcal{G}}$ of EL heterogeneous uncertain UAVs, a pinner with state $(q_0,\dot{q}_0)$, we want to find a distributed strategy for the inputs τ_i that respects the communication graph, that does not require knowledge of the EL matrices, and that leads to synchronization of the network, i.e. $[q_i,\dot{q}_i] \to [q_0,\dot{q}_0]$ as $t \to \infty$ for every UAV i. Let us start by formulating some reference dynamics:

TABLE III
Variables for Formation Control Law

Variable	Description
A_m, B_m	Reference dynamics
K_p, K_v	Reference gains
P	Lyapunov matrix
Γ	Adaptive gain
$\hat{D}_i, \hat{C}_i, \hat{g}_i$	Estimated dynamics of UAV # i
$\Theta_{Di}, \Theta_{Ci}, \Theta_{gi}$	Estimated gains of UAV # i
$\widehat{D_iD_{ji}}, D_i\widehat{D_jC_{ji}}$	Estimated dynamics between UAVs # i and # j
$\Theta_{D_iD_j}, \Theta_{D_iD_jD_j}$	Estimated gains between UAVs # i and # j

$$\begin{bmatrix}\dot{q}_0\\ \ddot{q}_0\end{bmatrix} = \underbrace{\begin{bmatrix}0 & \mathbb{I}\\ -K_p & -K_v\end{bmatrix}}_{A_m}\underbrace{\begin{bmatrix}q_0\\ \dot{q}_0\end{bmatrix}}_{x_m} + \underbrace{\begin{bmatrix}0\\ \mathbb{I}\end{bmatrix}}_{B_m} r \tag{8}$$

where $q_0, \dot{q}_0 \in \mathbb{R}^n$ is the state of the reference model and $r = \ddot{q}^d + K_v\dot{q}^d + K_pq^d$ is a user-specified reference input. The reference dynamics (8) basically represent some homogeneous dynamics all UAVs should synchronize to. With reference to the formation given in Fig. 9, we propose the controllers:

$$\begin{aligned}\tau_1 &= \underbrace{\Theta'_{D1}\xi_{D1}}_{\hat{D}_1}(-K_pq_1 - K_v\dot{q}_1 + r) + \underbrace{\Theta'_{C1}\xi_{C1}}_{\hat{C}_1}\dot{q}_1 + \underbrace{\Theta'_{g1}\xi_{g1}}_{\hat{g}_1}\\ \tau_2 &= \underbrace{\Theta'_{D2}\xi_{D2}}_{\hat{D}_2}(-K_pq_2 - K_v\dot{q}_2 + r) + \underbrace{\Theta'_{C2}\xi_{C2}}_{\hat{C}_2}\dot{q}_2 + \underbrace{\Theta'_{g2}\xi_{g2}}_{\hat{g}_2}\end{aligned} \tag{9}$$

where, the estimates $\hat{D}_1$, $\hat{C}_1$, $\hat{g}_1$ and $\hat{D}_2$, $\hat{C}_2$, $\hat{g}_2$ of the ideal matrices have been split in a linear-in-the-parameter form (i.e. any dynamic term is split as $\Theta'\xi(q,\dot{q})$ for some unknown parameter Θ and some known state-dependent regressor $\xi(q,\dot{q})$).

The adaptive laws for estimating such unknown Θ are:

$$\begin{aligned}\dot{\Theta}'_{C1} &= -\Gamma B'_mPe_1\dot{q}_1{}'\xi'_{C1}, \dot{\Theta}'_{g1} = -\Gamma B'_mPe_1\xi'_{g1}\\ \dot{\Theta}'_{D1} &= -\Gamma B'_mPe_1(-K_pq_1 - K_v\dot{q}_1 + r)'\xi'_{D1}\\ \dot{\Theta}'_{C2} &= -\Gamma B'_mPe_2\dot{q}_2{}'\xi'_{C1}, \dot{\Theta}'_{g2} = -\Gamma B'_mPe_2\xi'_{g2}\\ \dot{\Theta}'_{D2} &= -\Gamma B'_mPe_2(-K_pq_2 - K_v\dot{q}_2 + r)'\xi'_{D2}\end{aligned} \tag{10}$$

where, Γ is adaptive gain and $P = P' > 0$ is such that:

$$PA_m + A'_mP = -Q, \quad Q > 0. \tag{11}$$

The following controller is proposed for the other UAVs:

$$\begin{aligned}\tau_3 =& -\hat{D}_3[K_p(q_3 - q_1) + K_v(\dot{q}_3 - \dot{q}_1)] + \hat{C}_3\dot{q}_3\\ &+ \widehat{D_3D_1}\tau_1 - D_3\widehat{D_1C_1}\dot{q}_1 + \hat{g}_3\\ \tau_4 =& -\hat{D}_4[K_p(q_4 - q_2) + K_v(\dot{q}_4 - \dot{q}_2)] + \hat{C}_4\dot{q}_4\\ &+ \widehat{D_4D_2}\tau_2 - D_4\widehat{D_2C_2}\dot{q}_2 + \hat{g}_4\end{aligned} \tag{12}$$

Here, the adaptive laws for such an estimates are:

$$\begin{aligned}\dot{\Theta}'_{D_3D_1} &= -\Gamma B'_mPe_{13}\tau'_1\xi'_{D_3D_1}\\ \dot{\Theta}'_{D_3D_1C_1} &= -\Gamma B'_mPe_{13}\dot{q}'_1\xi'_{D_3D_1C_1}\\ \dot{\Theta}'_{C_3} &= -\Gamma B'_mPe_{13}\dot{q}'_3\xi'_{C_3}\\ \dot{\Theta}'_{g_3} &= -\Gamma B'_mPe_{13}\xi'_{g_3}\\ \dot{\Theta}'_{D3} &= -\Gamma B'_mPe_{13}[K_p(q_3 - q_1) + K_v(\dot{q}_3 - \dot{q}_1)'\xi'_{D_3D_1}]\end{aligned} \tag{13}$$

$$\dot{\Theta}'_{D_4D_2} = -\Gamma B'_m Pe_{24}\tau'_2\xi'_{D_4D_2}$$
$$\dot{\Theta}'_{D_4D_2C_2} = -\Gamma B'_m Pe_{24}\dot{q}'_2\xi'_{D_4D_2C_2}$$
$$\dot{\Theta}'_{C_4} = -\Gamma B'_m Pe_{24}\dot{q}'_4\xi'_{C_4}$$
$$\dot{\Theta}'_{g_4} = -\Gamma B'_m Pe_{24}\xi'_{g_4}$$
$$\dot{\Theta}'_{D4} = -\Gamma B'_m Pe_{24}[K_p(q_4 - q_2) + K_v(\dot{q}_4 - \dot{q}_2)'\xi'_{D_4D_2}]. \quad (14)$$

It is possible prove that, the proposed controllers and adaptive laws with all closed-loop signals are bounded, for any (i, j) such that $a_{ij} \neq 0$, we have $e_{ij} = (x_j - x_i) \rightarrow 0$ as $t \rightarrow \infty$. In addition, for every UAV j we have $e_j = (x_j - x_0) \rightarrow 0$ as $t \rightarrow \infty$. The proposed synchronization protocol can be extended to include gaps formation, provided that the error:

$$e_{ij} = x_j - x_i + v_{ji} = \begin{bmatrix} q_j \\ \dot{q}_j \end{bmatrix} - \begin{bmatrix} q_i \\ \dot{q}_i \end{bmatrix} + \begin{bmatrix} \bar{v}_{ji} \\ 0 \end{bmatrix} \quad (15)$$

is considered, where v_{ji} contains the desired formation displacement $\bar{v}_{ji}$ among UAVs j and i. In the forthcoming simulations we will consider the following parameters: constant airspeed $V_a = 15$ m/s, constant altitude $h_m = 50$ m. The control parameters of the vector field approach are $\kappa_{sl} = \kappa_o = \frac{\pi}{2}$, $k_{sl} = k_o = 0.1$, $\epsilon_{sl} = \epsilon_o = 1$, while the control parameters of the adaptive formation algorithm are

$$Q = 100\, I, \quad K_p = 50, \quad K_v = 50, \quad \Gamma = 100. \quad (16)$$

In line with most UAV path generation approaches, the path is composed of straight lines and orbits. For these simulations we take a path consisting of a straight line followed an orbit.

Fig. 10 shows the result of the simulations for an inverted V formation amongst the UAVs. The simulations of the multi-UAV formation are carried out for 4 UAVs and a pinner UAV. The communication graph shown in Fig. 9. It can be noted that the formation control task is achieved despite uncertainty, which demonstrates the effectiveness of the proposed formation control method. It must be remarked that the kinematic constraints of the UAV are not handled directly by the path following, but by the low level controllers (for pitch/roll/altitude) which are implemented inside ArduPilot. This implies that, for example, the radius of the circle path, which has been selected as 30 meters for all UAVs, should be decided according to physical limits: it cannot be too small otherwise the autopilot of the UAV would not be able to track the orbit (due to the maximum range of the aileron angle). More specifically, the following constraints are used in the model, in line with most commercial fixed-wing UAVs: the aileron command spans ±30 degrees, the elevator command spans ±15 degrees and the rudder command spans ±25 degrees. Table IV below shows the parameters of the fixed-wing UAVs, which are used only for the sake of simulations and are unknown for the purpose of control design. With respect to the initial conditions for the UAVs, the starting point can basically be arbitrary, and the initial attitude angles (pitch/roll/yaw) should be within the autopilot operating ranges, otherwise the autopilot will not be able to stabilize the UAV.

Remark 1: The benefit of the adaptive law is to allow all UAVs to homogenize to the same dynamics, by adapting the control action to compensate for different mass and inertia. In fact, it is well known in formation control literature that homogeneous dynamics are a crucial feature in order to achieve proper coordinated motion [19], [24].

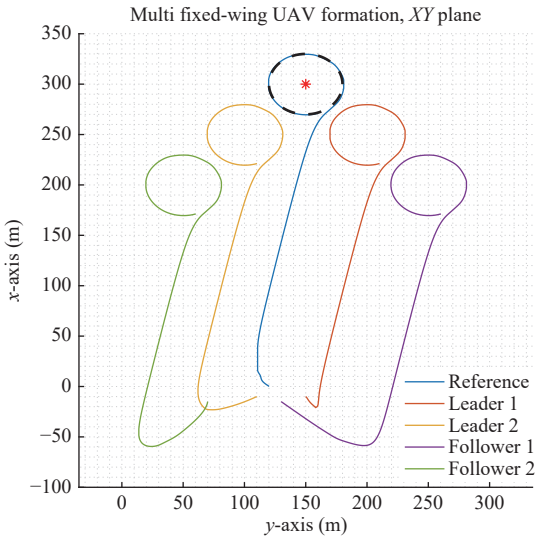

Fig. 10. Path following with V formation. The UAVs in the formation follow a line and then orbit around a point.

TABLE IV
FIXED-WING UAVS PARAMETERS

	Mass (kg)	Moment of Inertia (kgm^2)
UAV-0 (Pinner)	10	$I_x = 0.02$, $I_y = 0.026 I_z = 0.053$, $I_{xz} = 0.01$
UAV-1 (Leader 1)	20	$I_x = 0.1$, $I_y = 0.05 I_z = 0.1$, $I_{xz} = 0.01$
UAV-2 (Follower 1)	30	$I_x = 0.2$, $I_y = 0.1 I_z = 0.2$, $I_{xz} = 0.02$
UAV-3 (Leader 2)	40	$I_x = 0.4$, $I_y = 0.02 I_z = 0.4$, $I_{xz} = 0.04$
UAV-4 (Follower 2)	50	$I_x = 0.8$, $I_y = 0.04 I_z = 0.08$, $I_{xz} = 0.08$

The proposed algorithm can also be implemented with a different number of leaders and followers: Fig. 12 shows the result of the simulations for a Y formation (3 leaders and 1 follower) with control law

$$\tau_1 = \hat{D}_1(-K_p q_1 - K_v \dot{q}_1 + r) + \hat{C}_1 \dot{q}_1 + \hat{g}_1$$
$$\tau_2 = \hat{D}_2(-K_p q_2 - K_v \dot{q}_2 + r) + \hat{C}_2 \dot{q}_2 + \hat{g}_2$$
$$\tau_3 = \hat{D}_3(-K_p q_3 - K_v \dot{q}_3 + r) + \hat{C}_3 \dot{q}_3 + \hat{g}_3$$
$$\tau_4 = -\hat{D}_4[K_p(q_4 - q_3) + K_v(\dot{q}_4 - \dot{q}_3)] + \hat{C}_4 \dot{q}_4 + \widehat{D_4 D_3}\tau_3 - \widehat{D_4 D_3} C_3 \dot{q}_3 + \hat{g}_4 \quad (17)$$

with adaptive laws

$$\dot{\Theta}'_{C1} = -\Gamma B'_m Pe_1 \dot{q}_1{}' \xi'_{C1}, \dot{\Theta}'_{g1} = -\Gamma B'_m Pe_1 \xi'_{g1}$$
$$\dot{\Theta}'_{D1} = -\Gamma B'_m Pe_1(-K_p q_1 - K_v \dot{q}_1 + r)' \xi'_{D1}$$
$$\dot{\Theta}'_{C2} = -\Gamma B'_m Pe_2 \dot{q}_2{}' \xi'_{C2}, \dot{\Theta}'_{g2} = -\Gamma B'_m Pe_2 \xi'_{g2}$$
$$\dot{\Theta}'_{D2} = -\Gamma B'_m Pe_2(-K_p q_2 - K_v \dot{q}_2 + r)' \xi'_{D2}$$
$$\dot{\Theta}'_{C3} = -\Gamma B'_m Pe_3 \dot{q}_3{}' \xi'_{C3}, \dot{\Theta}'_{g3} = -\Gamma B'_m Pe_3 \xi'_{g3}$$
$$\dot{\Theta}'_{D3} = -\Gamma B'_m Pe_3(-K_p q_3 - K_v \dot{q}_3 + r)' \xi'_{D3}$$
$$\dot{\Theta}'_{D_4D_3} = -\Gamma B'_m Pe_{34}\tau'_3 \xi'_{D_4D_3}$$
$$\dot{\Theta}'_{D_4D_3C_3} = -\Gamma B'_m Pe_{34}\dot{q}'_3 \xi'_{D_4D_3C_3}$$
$$\dot{\Theta}'_{C_4} = -\Gamma B'_m Pe_{34}\dot{q}'_4 \xi'_{C_4}, \dot{\Theta}'_{g_4} = -\Gamma B'_m Pe_{34}\xi'_{g_4}$$
$$\dot{\Theta}'_{D4} = -\Gamma B'_m Pe_{34}[K_p(q_4 - q_3) + K_v(\dot{q}_4 - \dot{q}_3)' \xi'_{D_4D_3}]. \quad (18)$$

In other words, the structure of the controller is suitable for any formation, but because each UAVs has different neighbors according to the formation, the signals used to implement the control action will be different. The communication graph for the Y formation is shown in Fig. 11. Fig. 14 shows the result of the simulations for an inverted T formation (1 leader and 3 followers) with control law

$$\begin{aligned}
\tau_1 &= \hat{D}_1(-K_p q_1 - K_v \dot{q}_1 + r) + \hat{C}_1 \dot{q}_1 + \hat{g}_1 \\
\tau_2 &= -\hat{D}_2[K_p(q_2 - q_1) + K_v(\dot{q}_2 - \dot{q}_1)] + \hat{C}_2 \dot{q}_2 \\
&\quad + \widehat{D_2 D_1}\tau_1 - D_2\widehat{D_1 C_1}\dot{q}_1 + \hat{g}_2 \\
\tau_3 &= -\hat{D}_3[K_p(q_3 - q_1) + K_v(\dot{q}_3 - \dot{q}_1)] + \hat{C}_3 \dot{q}_3 \\
&\quad + \widehat{D_3 D_1}\tau_1 - D_3\widehat{D_1 C_1}\dot{q}_1 + \hat{g}_3 \\
\tau_4 &= -\hat{D}_4[K_p(q_4 - q_1) + K_v(\dot{q}_4 - \dot{q}_1)] + \hat{C}_4 \dot{q}_4 \\
&\quad + \widehat{D_4 D_1}\tau_1 - D_4\widehat{D_1 C_1}\dot{q}_1 + \hat{g}_4
\end{aligned} \tag{19}$$

with adaptive laws

$$\begin{aligned}
&\dot{\Theta}'_{C1} = -\Gamma B'_m P e_1 \dot{q}_1{}' \xi'_{C1}, \dot{\Theta}'_{g1} = -\Gamma B'_m P e_1 \xi'_{g1} \\
&\dot{\Theta}'_{D1} = -\Gamma B'_m P e_1 (-K_p q_1 - K_v \dot{q}_1 + r)' \xi'_{D1} \\
&\dot{\Theta}'_{D_2 D_1} = -\Gamma B'_m P e_{12} \tau'_1 \xi'_{D_2 D_1} \\
&\dot{\Theta}'_{D_2 D_1 C_1} = -\Gamma B'_m P e_{12} \dot{q}'_1 \xi'_{D_2 D_1 C_1} \\
&\dot{\Theta}'_{C_2} = -\Gamma B'_m P e_{12} \dot{q}'_2 \xi'_{C_2}, \dot{\Theta}'_{g_2} = -\Gamma B'_m P e_{12} \xi'_{g_2} \\
&\dot{\Theta}'_{D2} = -\Gamma B'_m P e_{12} [K_p(q_2 - q_1) \\
&\qquad + K_v(\dot{q}_2 - \dot{q}_1)' \xi'_{D_2 D_1}]
\end{aligned} \tag{20}$$

$$\begin{aligned}
&\dot{\Theta}'_{D_3 D_1} = -\Gamma B'_m P e_{13} \tau'_1 \xi'_{D_3 D_1} \\
&\dot{\Theta}'_{D_3 D_1 C_1} = -\Gamma B'_m P e_{13} \dot{q}'_1 \xi'_{D_3 D_1 C_1} \\
&\dot{\Theta}'_{C_3} = -\Gamma B'_m P e_{13} \dot{q}'_3 \xi'_{C_3}, \dot{\Theta}'_{g_3} = -\Gamma B'_m P e_{13} \xi'_{g_3} \\
&\dot{\Theta}'_{D3} = -\Gamma B'_m P e_{13} [K_p(q_3 - q_1) \\
&\qquad + K_v(\dot{q}_3 - \dot{q}_1)' \xi'_{D_3 D_1}]
\end{aligned} \tag{21}$$

$$\begin{aligned}
&\dot{\Theta}'_{D_4 D_1} = -\Gamma B'_m P e_{14} \tau'_1 \xi'_{D_4 D_1} \\
&\dot{\Theta}'_{D_4 D_1 C_1} = -\Gamma B'_m P e_{14} \dot{q}'_1 \xi'_{D_4 D_1 C_1} \\
&\dot{\Theta}'_{C_4} = -\Gamma B'_m P e_{14} \dot{q}'_4 \xi'_{C_4}, \dot{\Theta}'_{g_4} = -\Gamma B'_m P e_{14} \xi'_{g_4} \\
&\dot{\Theta}'_{D4} = -\Gamma B'_m P e_{14} [K_p(q_4 - q_1) \\
&\qquad + K_v(\dot{q}_4 - \dot{q}_1)' \xi'_{D_4 D_1}].
\end{aligned} \tag{22}$$

The communication graph for the inverted T formation is shown in Fig. 13.

C. The Importance of Adaptation

Finally, we would like to highlight the relevance of embedding adaptation in formation control by showing what happens in the absence of such adaptation. To this purpose, we set up another simulation with inverted V formation in which two UAVs (Leader 2 and Follower 2) adopt the adaptive algorithm, whereas the other two (Leader 1 and Follower 1) do not employ adaptation. This means that their control gains are kept fixed without adapting to different mass/inertia. Fig. 15 shows the result of such simulation: it can be seen that the two UAVs not employing adaptation cannot close the gap with respect to their predecessor and they eventually leave the formation. It can be noted from Table IV

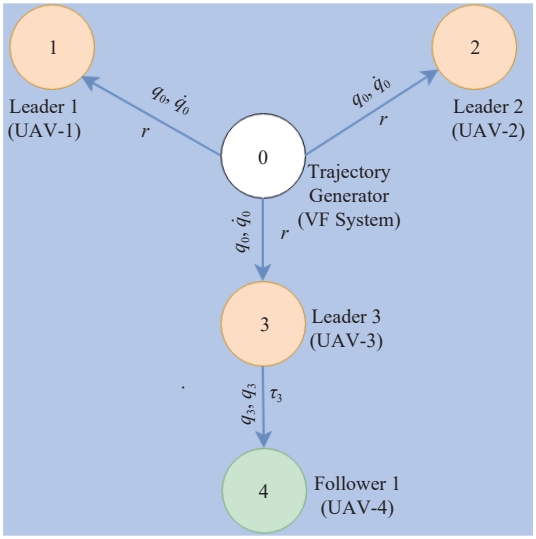

Fig. 11. Communication graph with Y formation.

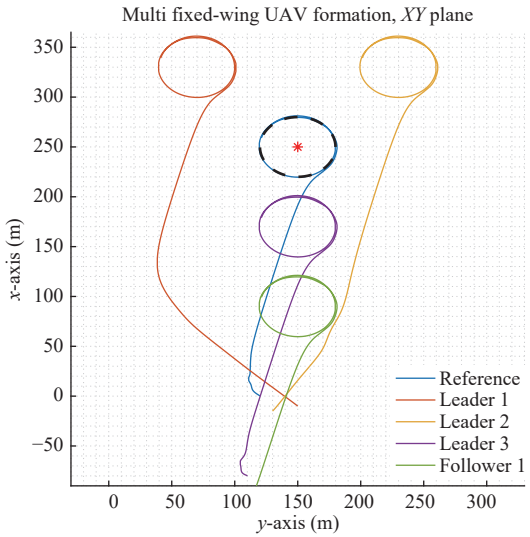

Fig. 12. Path following with Y formation. The UAVs in the formation follow a line and then orbit around a point.

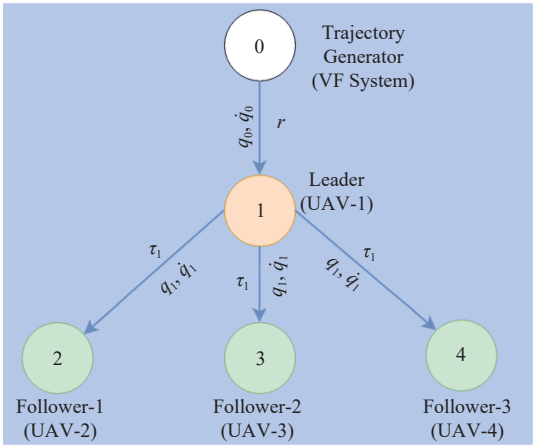

Fig. 13. Communication graph with T formation.

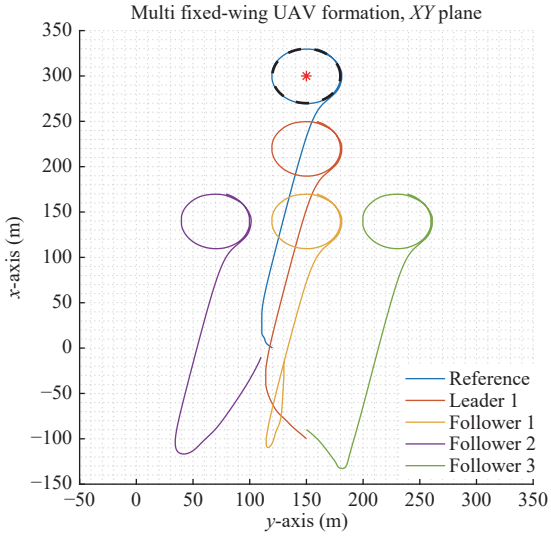

Fig. 14. Path following with T formation. The UAVs in the formation follow a line and then orbit around a point.

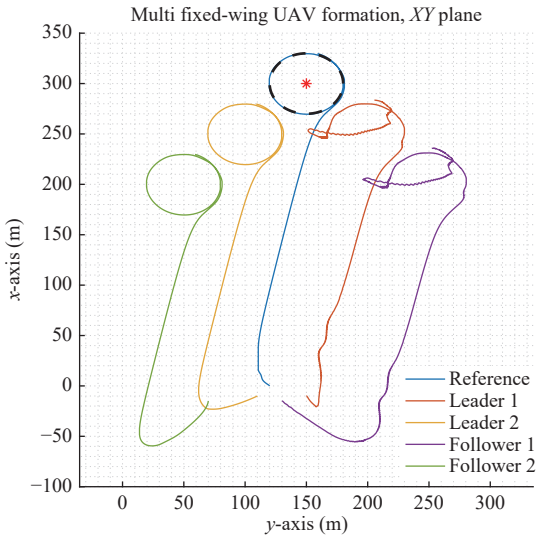

Fig. 15. Unsuccessful path following in the absence of adaptation. Leader 2 and Follower 2, employing the adaptive algorithm, manage to achieve their part of the formation, while Leader 1 and Follower 1, which do not employ adaptation, leave the formation.

that the masses of the UAVs vary of a factor 5, whereas the inertia vary of a factor 10: it is remarkable that a unique algorithm can adapt to such heterogeneity. In the absence of such adaptation, it might be difficult to find a formation control strategy that can work for any inertia and mass. Therefore, the proposed software-in-the-loop simulations show the capability of achieving different UAV formations while handling uncertain mass and inertia.

VII. Conclusions

The paper has discussed the research activities on the design and software-in-the-loop implementation of adaptive formation controllers for fixed-wing unmanned aerial vehicles (UAVs). The focus of this paper was on the control and simulation of fixed-wing UAVs in Matlab environment, in the presence of parametric uncertainties represented by uncertain mass and inertia. Several aspects of the guidance and control for fixed-wing UAVs have been tackled: Matlab modelling of UAVs, hardware and software integration, ArduPilot autopilot low-level (roll/pitch/altitude) control, vector field path following, adaptive formation control and finally the software-in-the-loop simulations. Software-in-the-loop capability was achieved by replicating in Matlab the ArduPilot code (according to the Ardupilot documentation and to the Ardupilot code itself). This reversed engineering step allowed us to perform simulations with the actual autopilot protocols of the UAV. Future work will cover hardware-in-the-loop simulations (the actual flight controller will send commands and receive measures from the Matlab simulator), as well as the real flights.

References

[1] H. Chao, Y. Cao, and Y. Chen, "Autopilots for small unmanned aerial vehicles: a survey," *International Journal of Control, Automation and Systems*, vol. 8, no. 1, pp. 36–44, 2010.

[2] A. Isidori, L. Marconi, and A. Serrani, *Robust autonomous guidance: an internal model approach*. Springer Science & Business Media, 2012.

[3] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano. A. Sala, and N. Tomatis, "The SHERPA project: smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments, " in *Proc. 2012 IEEE Int. Symposium on Safety, Security, and Rescue Robotics* (*SSRR*), 2012, pp. 1–4.

[4] P. B. Sujit, S. Saripalli, and J. B. Sousa, "Unmanned aerial vehicle path following: a survey and analysis of algorithms for fixed-wing unmanned aerial vehicles," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, 2014.

[5] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic, "Performance limitations in reference tracking and path following for nonlinear systems," *Automatica*, vol. 44, no. 3, pp. 598–610, 2008.

[6] L. Furieri, T. Stastny, L. Marconi, R. Siegwart, and I. Gilitschenski, "Gone with the wind: nonlinear guidance for small fixed-wing aircraft in arbitrarily strong windfields, " in *Proc. 2017 American Control Conf.* (*ACC'17*), pp. 4254–4261.

[7] D. Invernizzi and M. Lovera, "Trajectory tracking control of thrustvectoring UAVs," *Automatica*, vol. 95, pp. 180–186, 2018.

[8] D. V. Dimarogonas, "Sufficient conditions for decentralized potential functions based controllers using canonical vector fields," *IEEE Transactions on Automatic Control*, vol. 57, no. 10, pp. 2621–2626, 2012.

[9] M. Kothari, I. Postlethwaite, and D.-W. Gu, "UAV path following in windy urban environments," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 1013–1028, 2014.

[10] F. Gavilan, R. Vazquez, and S. Esteban, "Trajectory tracking for fixedwing UAV using model predictive control and adaptive backstepping," in *Proc. 1st IFAC Workshop on Advanced Control and Navigation for Autonomous Aerospace Vehicles (ACNAAV'15)*, pp. 132-137, pp. 132–137, 2015.

[11] J. Chang, J. Cieslak, J. Dávila, A. Zolghadri, and J. Zhou, "Analysis and design of second-order sliding-mode algorithms for quadrotor roll and pitch estimation," *ISA Trans.*, pp. 495–512, 2017.

[12] G. Casadei, L. Furieri, N. Mimmo, R. Naldi, and L. Marconi, "Internal model-based control for loitering maneuvers of UAVs, " in *Proc. 2016 European Control Conf.* (*ECC*), pp. 672–677.

[13] J. Chang, J. Cieslak, J. Davila, J. Zhou, A. Zolghadri, and Z. Guo, "A two-step approach for an enhanced quadrotor attitude estimation via imu data," *IEEE Trans. on Control Systems Technology*, vol. 26, no. 3, pp. 1140–1148, 2018.

[14] B. Zhou, H. Satyavada, and S. Baldi, "Adaptive path following for unmanned aerial vehicles in time-varying unknown wind environment," in *Proc. American Control Conf. (ACC'17)*, pp. 1127–1132, 2017.

[15] N. Cho and Y. Kim, "Three-Dimensional nonlinear differential geometric path-following guidance law," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 12, pp. 948–954, 2015.

[16] H. Chen, K. Chang, and C. S. Agate, "UAV path planning with tangentplus-lyapunov vector field guidance and obstacle avoidance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 840–856, 2013.

[17] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 519–529, 2007.

[18] S. Baldi, S. Yuan, and P. Frasca, "Output synchronization of unknown heterogeneous agents via distributed model reference adaptation," *IEEE Transactions on Control of Network Systems*, 2018.

[19] S. Baldi and P. Frasca, "Adaptive synchronization of unknown heterogeneous agents: an adaptive virtual model reference approach," *Journal of the Franklin Institute*, vol. 356, no. 2, pp. 935–955, 2019.

[20] S. Baldi, "Cooperative output regulation of heterogeneous unknown systems via passification-based adaptation," *IEEE Control Systems Letters*, vol. 2, no. 1, pp. 151–156, 2018.

[21] Y. Abou Harfouch, S. Yuan, and S. Baldi, "An adaptive switched control approach to heterogeneous platooning with inter-vehicle communication losses," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1434–1444, 2018.

[22] S. Baldi, M. R. Rosa, and P. Frasca, "Adaptive state-feedback synchronization with distributed input: the cyclic case," in *Proc. 7th IFAC Workshop on Distributed Estimation and Control in Networked Systems (NECSYS), Groningen, The Netherlands*, 2018.

[23] S. Baldi, I. A. Azzollini, and E. B. Kosmatopoulos, "A distributed disagreement-based protocol for synchronization of uncertain heterogeneous agents," *European Control Conf.*, *Limassol*, *Cyprus*, 2018.

[24] Y. Abou Harfouch, S. Yuan, and S. Baldi, "An adaptive switched control approach to heterogeneous platooning with inter-vehicle communication losses," in *Proc. 20th IFAC World Congr., Toulouse, France*, pp. 1382–1387, 2017.

[25] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. John Wiley & Sons, 2015.

[26] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[27] "Dryden wind turbulence model (discrete) Simulink, " 2019. [Online]. Available: https://nl.mathworks.com/help/aeroblks/wind.html

[28] "Aerospace block-set Simulink, " 2019. [Online]. Available: https://nl.mathworks.com/help/aeroblks/index.html

[29] S. Farì, "Guidance and control for a fixed-wing UAV, " M.S. thesis, POLIMI. , IT, 2017.

[30] "Ardupilot documentation, " 2019. [Online]. Available: http://ardupilot.org/

[31] "Learning the ardupilot codebase, " 2019. [Online]. Available: http://ardupilot.org/dev/docs/learning-the-ardupilot-codebase.html

[32] "Roll, pitch and yaw controller tuning, " 2019. [Online]. Available: http://ardupilot.org/plane/docs/roll-pitch-controller-tuning.html

[33] L. Meier, D. Honegger, and M. Pollefeys, "Px4: a node-based multithreaded open source robotics framework for deeply embedded platforms, " in *Proc. 2015 IEEE Int. Conf. on Robotics and Automation* (*ICRA*), pp. 6235–6240.

[34] "Hkpilot32 flight controller, " 2019. [Online]. Available: https://docs.px4.io/en/flight-controller/HKPilot32.html

Jun Yang received the MS degree in electronic information engineering from Northwestern Polytechnical university. He is a Research Fellow and a System Designer at the Systems Engineering Research Institute of China. His current research interests include unmanned air vehicle systems, airborne detection information processing, and object detection and tracking.

Ximan Wang received the B.Sc. degree from Taiyuan University, in 2014, and the M.Sc. degree from University of Sheffield, in 2016. He was a Senior Engineer at Systems Engineering Research Institute, Beijing, China, and he is now pursuing the Ph.D. degree at the Delft Center for Systems and Control, Delft University of Technology with research interests in adaptive optimization for control and UAV adaptive control.

Simone Baldi received the B.Sc. degree in electrical engineering, and the M.Sc. and Ph.D. degrees in automatic control systems engineering from the University of Florence, Italy, in 2005, 2007, and 2011, respectively. He is currently Professor at the School of Mathematics, Southeast University, with a guest position at the Delft Center for Systems and Control, Delft University of Technology, where he was assistant professor. Previously, he held postdoctoral researcher positions at the University of Cyprus, and at the Information Technologies Institute, Centre for Research and Technology Hellas. His research interests include adaptive systems and switching control with applications in networked control systems and multi-agent systems.

Satish Singh received the B.E. (Bachelor of Engineering) degree in electrical engineering from Nagpur University, India in 2012. He is currently pursuing his M.Sc. degree in embedded systems from Delft University of Technology, Delft, The Netherlands. His work focuses on software-in-the-loop and hardware-in-theloop simulations for UAVs.

Stefano Farì received the B.Sc. and M.Sc. in automation engineering from Politecnico di Milano, Italy, performing his master thesis as guest researcher at the Delft Center for Systems and Control, Delft University of Technology, The Netherlands. He has worked at Piaggio Aerospace, Savona, Italy, as flight control system engineer and he is now working as GNC engineer at German Aerospace Center (DLR), Bremen, Germany.