ArduPilot-Based Adaptive Autopilot
Architecture and Software-in-The-Loop Experiments

Baldi, Simone; Sun, Danping; Xia, Xin; Zhou, Guopeng; Liu, Di

DOI
10.1109/TAES.2022.3162179

Publication date
2022

Document Version
Final published version

Published in
IEEE Transactions on Aerospace and Electronic Systems

ArduPilot-Based Adaptive Autopilot: Architecture and Software-in-the-Loop Experiments

SIMONE BALDI, Senior Member, IEEE

DANPING SUN

XIN XIA, Graduate Student Member, IEEE

GUOPENG ZHOU

DI LIU, Member, IEEE

This article presents an adaptive method for ArduPilot-based autopilots of fixed-wing unmanned aerial vehicles (UAVs). ArduPilot is a popular open-source unmanned vehicle software suite. We explore how to augment the PID loops embedded inside ArduPilot with a model-free adaptive control method. The adaptive augmentation, adopted for both attitude and total energy control, uses input/output data without requiring an explicit model of the UAV. The augmented architecture is tested in a software-in-the-loop UAV platform in the presence of several uncertainties (unmodeled low-level dynamics, different payloads, time-varying wind, and changing mass). The performance is measured in terms of tracking errors and control efforts of the attitude and total energy control loops. Extensive experiments with the original ArduPilot, the proposed augmentation, and alternative autopilot strategies show that the augmentation can significantly improve the performance for all payloads and wind conditions: the UAV is less affected by wind and exhibits more than 70% improved tracking, with more than 7% reduced control effort.

Manuscript received 23 August 2021; revised 23 November 2021 and 22 January 2022; released for publication 21 March 2022. Date of publication 24 March 2022; date of current version 11 October 2022.

DOI. No. 10.1109/TAES.2022.3162179

Refereeing of this contribution was handled by G. Inalhan.

This work was supported in part by the Natural Science Foundation of China under Grant 62073074, in part by the Special Funding for Overseas under Grant 6207011901, in part by the Research Fund for International Scientists under Grant 62150610499, in part by the Double Innovation Plan under Grant 4207012004, and in part by the Science and Technology Plan Project of Hubei Province (second batch) under Grant 2019BEC206.

Authors' addresses: Simone Baldi is with the School of Mathematics, Southeast University, Nanjing 210096, China and also guest with Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands, E-mail: (s.baldi@tudelft.nl); Danping Sun is with the Wuhan Textile University, Wuhan 430200, China, and also with the Hubei Electrical Machinery and Control System Engineering Technology Research Center, Xianning 437100, China, E-mail: (sunsundp24@163.com); Xin Xia is with the School of Mathematics, Southeast University, Nanjing 211189, China, E-mail: (xiaxin0209@gmail.com); Guopeng Zhou is with the Hubei Electrical Machinery and Control System Engineering Technology Research Center, Xianning 437100, China, and also with Hubei University of Science and Technology, Xianning 437100, China, E-mail: (zhgpeng@hbust.edu.cn); Di Liu is with the School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China and also with the Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen (RUG), 9712 CP Groningen, The Netherlands, E-mail: (liud923@126.com). *(Corresponding authors: Di Liu; Guopeng Zhou.)*

I. INTRODUCTION

The term "autopilot" refers to a system used to control the trajectory of an aircraft, marine craft, or self-driving car without requiring constant manual control by a human operator. Historically, when talking about autopilots, researchers have addressed missile and ship autopilots: representative examples are model reference sliding surface for missile autopilot [1], adaptive compensation for backlash hysteresis [2], two-loop and three-loop adaptive missile autopilots [3], [4], L_1 adaptive control augmenting a dynamic inversion missile autopilot [5], ship autopilot for time-varying control coefficients [6], adaptive fuzzy control for ship autopilot [7], [8], and many more. This list, although nonexhaustive, explains how often the term "adaptive" is associated to autopilots, due to the presence of uncertainties in the vehicle dynamics and in the environment. With the development of unmanned aerial vehicle (UAV) technology, many researchers have turned their attention toward the design of autopilots for UAVs. While missiles, ships, and UAVs all face uncertainties [9], their specific characteristics require appropriate designs: several control methods applied to UAVs are model-based, such as robust [10] and optimal control [11]. Model-based methods, however, rely on precise mathematical models of the UAV and of the environment. But in real life, it is difficult to obtain an accurate model that takes into account environmental influences on the UAV dynamics. Therefore, model-based control methods should be augmented or replaced by adaptive [12]–[14] or intelligent control [15] methods, to handle some uncertainty.

Many approaches to UAV autopilot deal with rotor UAVs (cf., [16]–[20]). At the same time, the flight control community has turned its attention toward fixed-wing UAVs, which often require approaches similar to large-scale airplanes. Examples include autopilots adapting to icing [21], to degradation of the aerodynamics [22], and to reduction of the control effectiveness [23]. It is well-recognized that control of fixed-wing UAVs requires simplifying assumptions in the autopilot design: the most typical simplification is assuming that roll/pitch/yaw and velocity dynamics do not interact with each other [24]. Off-the-shelf open-source autopilots (Pixhawk, ArduPilot, NAVIO, etc., [25], [26]) also design roll/pitch/yaw and velocity control loops independently, using PID controllers with rate and

position feedback. This approach is suggested in most textbooks [27], [28]. Although the PID approach is adequate in standard situations, there are scenarios where the aerodynamics effects and mass/inertia changes will dramatically reduce the effectiveness of these traditional autopilot laws. Under such conditions, robustness to unmodeled dynamics or adaptation to uncertain dynamics need to be embedded in the control [29]. The review of intelligent autopilots for UAVs in [15] clarifies research opportunities associated to the design of adaptation in autopilots, which can take two routes: the first one is to define a completely new architecture useful to embed appropriate adaptation laws [30]; the second one is to integrate adaptation in the established open-source architectures to guarantee higher acceptance from the UAV community.

Considering the higher chances of acceptance, this second route stimulates a research toward making existing autopilot architectures adaptive. A reasonable approach to accomplish this goal is the model-free adaptive control (MFAC), originally proposed by Hou *et al.* [31], [32]. The model-free adaptive control method has been widely used in industrial applications, such as synchronous machines [33], spacecrafts [34], heading control [35], flapping wing control [36], 6WID/4WIS navigation for ground vehicles [37], [38]. The basic idea of model-free adaptive control is to establish a dynamic linear model of the nonlinear system at the current operating point: this is done using input/output data of the controlled system to estimate a set of pseudo partial derivatives. These derivatives are used to optimize a cost via a one-step ahead controller. As such, model-free adaptive control belongs to the large family of data-driven control methods that do not rely on the knowledge of the system model. Other representative methods in this family are iterative feedback tuning [39], virtual reference feedback tuning [40], unfalsified control [41], and many more.

Summarizing, while this work on the one hand can be classified as an application of model-free adaptive control, on the other hand it has distinct contributions.

1) To the best of our knowledge, it is the first time that model-free adaptive control is integrated in a complete ArduPilot-based autopilot. This means that the original ArduPilot architecture (developed and maintained by a huge community) is not modified. This can increase acceptance toward the adoption of this method;
2) The model-free adaptive control method is integrated in the low level roll/pitch/yaw control and in the total energy control system (TECS) of ArduPilot. We are not aware of a similar implementation in the literature. By using different combinations of original and augmented loops, we show that the model-free adaptive control augmentation is always beneficial for any loop of the ArduPilot architecture.
3) Realistic software-in-the-loop experiments and comparisons with alternative autopilots validate the architecture in the presence of several uncertainties (unmodeled low-level dynamics, different payloads, time-varying wind, and changing mass). ArduPilot functionalities are emulated according to the ArduPilot documentation and code [42]. The augmentation method significantly improves the performance for all payloads and wind conditions: the UAV exhibits more than 70% improved tracking, with more than 7% reduced control effort.

The rest of this article is organized as follows. Preliminaries on ArduPilot architecture are in Section II. The model-free adaptive control is recalled in Section III. The integration of the ArduPilot architecture and model-free adaptive control is presented in Section IV. Section V gives the software-in-the-loop experiments. Finally, Section VI concludes this article. The Appendix gives basic information on UAV dynamics.

II. PRELIMINARIES ON ARDUPILOT ARCHITECTURE

Recalling some principles underlying the ArduPilot architecture will help understanding how such architecture can be augmented with adaptation capabilities.

A. Roll/Pitch/Yaw Linear Design Models

Because the ArduPilot architecture is based on PID, it is useful to recall how appropriate linear models can be obtained for roll/pitch/yaw. To this purpose, dynamics of a fixed-wing are decomposed into lateral motion (roll and course angle) and longitudinal motion (airspeed, pitch angle, and altitude).

For lateral dynamics, the aileron angle δ_{a} is primarily used to influence the roll rate p, while the rudder angle δ_{r} is used to control the yaw ψ. Under simplifying assumptions (the interested reader can check [27, Ch. 6]), one obtains second-order dynamics between the aileron and the roll angle

$$\phi(s) = \frac{\alpha_{\phi_1}}{s(s+\alpha_{\phi_2})}\left(\delta_{\mathrm{a}}(s) + \frac{1}{\alpha_{\phi_2}} d_{\phi_2}(s)\right) \tag{1}$$

where s is the Laplace operator, $\alpha_{\phi_1}, \alpha_{\phi_2}$ are coefficients coming from linearization, and d_{ϕ_2} is a disturbance accounting for unmodeled dynamics. Furthermore, one obtains first-order dynamics between the roll and the course angle

$$\chi(s) = \frac{g/V_{\mathrm{g}}}{s}(\phi(s) + d_{\chi}(s)) \tag{2}$$

where χ is the course angle, V_{g} is the ground speed, and d_{χ} is another disturbance. This cascade of second-order and first-order dynamics can be used to design a cascaded PID loop, called low-level roll control. For the side slip, one obtains first-order dynamics between the rudder and the side slip angle β_{sa}

$$\beta_{\mathrm{sa}}(s) = \frac{\alpha_{\beta_2}}{s+\alpha_{\beta_1}}(\delta_{\mathrm{r}}(s) + d_{\beta}(s)) \tag{3}$$

where $\alpha_{\beta_1}, \alpha_{\beta_2}$ are coefficients coming from linearization, and d_{β} is a disturbance accounting for unmodeled dynamics. These first-order dynamics can be used to design another PID loop, called low-level yaw control.

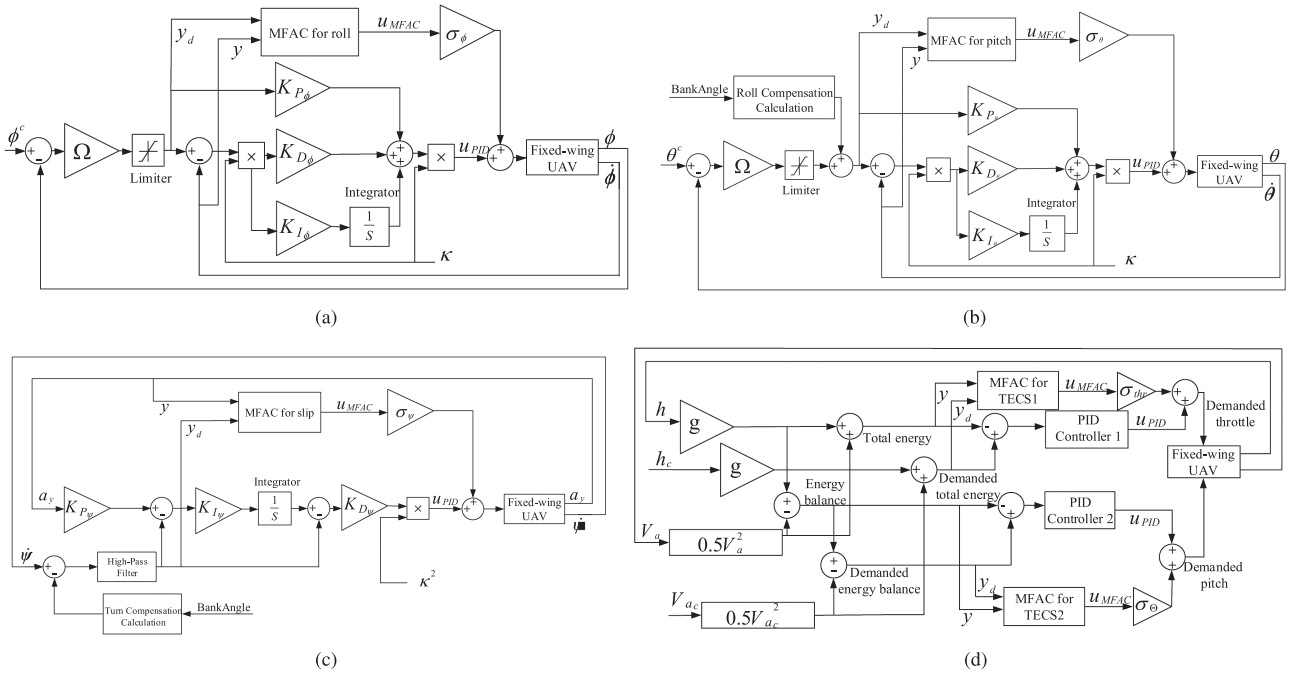


Fig. 1. ArduPilot schemes for all control loops. In this work, each loop is augmented by an adaptive module that uses appropriate measurements: angle and desired (filtered) angle for roll/pitch/yaw, total energy, and energy balance for TECS. (a) Roll control loop. (b) Pitch control loop. (c) Side-slip control. (d) Total energy control system loop.

For the longitudinal dynamics, the controls are the elevator angle δ_e and the throttle percentage δ_t (percentage of the maximum throttle). The elevator angle is used to influence the pitch angle θ, while the pitch angle is used to manipulate both the altitude h and the airspeed V_a. Simplifying assumptions (the interested reader can check [27, Ch. 5]) give second-order dynamics between the elevator and the pitch angle

$$\theta(s) = \frac{\alpha_{\theta_3}}{s^2 + \alpha_{\theta_1} s + \alpha_{\theta_2}} \left(\delta_e(s) + \frac{1}{\alpha_{\theta_3}} d_{\theta_2}(s) \right) \tag{4}$$

where $\alpha_{\theta_1}, \alpha_{\theta_2}, \alpha_{\theta_3}$ are coefficients coming from linearization, and d_{θ_2} is a disturbance accounting for unmodeled dynamics. These second-order dynamics can be used to design a cascaded PID loop, called low-level pitch control.

B. Cascaded Control

In ArduPilot, the term "cascaded control" (also called successive loop closure) refers to closing PID loops in succession around first-order or second-order system dynamics. The cascaded control of ArduPilot uses some scaling factors: for example, a scaling is introduced to moderate the wing deflections according to the airspeed

$$\kappa = \frac{V_{a,\text{nom}}}{V_a} \tag{5}$$

where $V_{a,\text{nom}}$ is a nominal cruise speed (by default $V_{a,\text{nom}} = 15$m/s). The scaling factor κ represents the fact that the wing surfaces should be moved less at high speed, and more at low speed. The low-level roll control is

$$u_\phi(t) = \kappa K_{P_\phi} y_{d_\phi}(t) + \kappa^2 K_{I_\phi} \int_{t_0}^{t} (y_{d_\phi}(\tau) - \phi(\tau)) d\tau + \kappa^2 K_{D_\phi} (y_{d_\phi}(t) - \phi(t)) \tag{6}$$

where $(K_{P_\phi}, K_{I_\phi}, K_{D_\phi})$ are PID gains, $y_{d_\phi} = \text{LIM}((\phi^c - \phi)\Omega_\phi)$, and LIM refers to a limiter function (saturation)

$$\text{LIM}(e_\phi) = \begin{cases} \overline{e}_\phi \text{ if } e_\phi \geq \overline{e}_\phi \\ e_\phi \text{ if } \underline{e}_\phi < e_\phi < \overline{e}_\phi \\ \underline{e}_\phi \text{ if } e_\phi \leq \underline{e}_\phi \end{cases} \tag{7}$$

where $\overline{e}_\phi = -\underline{e}_\phi = 75^\circ$. The roll control scheme is represented in Fig. 1(a). The low-level pitch control is

$$u_\theta(t) = \kappa K_{P_\theta} y_{d_\theta}(t) + \kappa^2 K_{I_\theta} \int_{t_0}^{t} (y_{d_\theta}(\tau) - \theta(\tau)) d\tau + \kappa^2 K_{D_\theta} (y_{d_\theta}(t) - \theta(t)) \tag{8}$$

where $(K_{P_\theta}, K_{I_\theta}, K_{D_\theta})$ are PID gains, $y_{d_\theta} = \text{LIM}((\theta^c - \theta)\Omega) + \theta_{\text{bank}}$, LIM refers to another limiter function

$$\text{LIM}(e_\theta) = \begin{cases} \overline{e}_\theta \text{ if } e_\theta \geq \overline{e}_\theta \\ e_\theta \text{ if } \underline{e}_\theta < e_\theta < \overline{e}_\theta \\ \underline{e}_\theta \text{ if } e_\theta \leq \underline{e}_\theta \end{cases} \tag{9}$$

where $\overline{e}_\theta = -\underline{e}_\theta = 75^\circ$, and θ_{bank} refers to the bank angle for roll compensation

$$\theta_{\text{bank}} = \frac{g}{V_a} |\tan\theta \sin\theta| \cos\theta. \tag{10}$$

The pitch control scheme is given in Fig. 1(b). The low-level yaw control is

$$u_\psi(t) = \kappa^2 \left(K_{\mathrm{I}_\psi} \int_{t_0}^{t} (K_{\mathrm{P}_\psi} a_y(\tau) - y_{d_\psi}(\tau)) d\tau - y_{d_\psi}(t) \right) K_{\mathrm{D}_\psi} \tag{11}$$

where $(K_{\mathrm{P}_\psi}, K_{\mathrm{I}_\psi}, K_{\mathrm{D}_\psi})$ are PID gains, $y_{d_\psi} = H(s)(\dot{\psi} - \dot{\psi}_{\mathrm{turn}})$, $\dot{\psi}_{\mathrm{turn}}$ refers to the compensation for turn coordination

$$\dot{\psi}_{\mathrm{turn}} = \frac{g}{V_{\mathrm{a}}} \tan\phi \cos\phi \tag{12}$$

and $H(s)$ is a high-pass filter with cutoff frequency at 0.2 rd/s

$$H(s) = \frac{s}{s + 0.2}. \tag{13}$$

The side-slip control scheme is given in Fig. 1(c).

C. Total Energy Control System

Although the airspeed can be controlled by means of throttle δ_{t} and the altitude by means of elevator δ_{e}, altitude and airspeed dynamics are not decoupled: for example, for a constant thrust, the airspeed is affected by the UAV pitching up or down, since pitching allows to convert some of the kinetic energy into potential energy. Motivated by this scenario, the ArduPilot developers have proposed a control design based on energy considerations, called TECS. Consider the standard definitions for kinetic energy $E_{\mathrm{k}} \triangleq \frac{1}{2} V_{\mathrm{a}}^2$ and potential energy $E_{\mathrm{p}} \triangleq gh$ (the convention of ArduPilot is that the mass does not appear in the definitions). Accordingly, the energy rates are $\dot{E}_{\mathrm{k}} \triangleq V_{\mathrm{a}} \dot{V}_{\mathrm{a}}$ and $\dot{E}_{\mathrm{p}} \triangleq g\dot{h}$. Note that $\dot{V}_{\mathrm{a}}$ can be obtained from the on-board accelerometers. On the other hand, an approximate $\dot{h}$ can be obtained as follows: in absence of wind, the angle between V_{g} and the horizontal plane, is the flight path angle γ_{fp}. So

$$\dot{h} = V_{\mathrm{g}} \sin\gamma_{\mathrm{fp}} \tag{14}$$

where V_{g} indicates the ground speed. One can define the commanded airspeed and altitude ($V_{\mathrm{a,c}}$ and h_{c}) in terms of their energy, i.e.,

$$E_{\mathrm{k,c}} = \frac{1}{2} V_{\mathrm{a,c}}^2, \quad E_{\mathrm{p,c}} = gh_{\mathrm{c}} \tag{15}$$

and the commanded energy rates follow accordingly. The total energy and the energy difference can be defined as

$$E_{\mathrm{T}} = E_{\mathrm{k}} + E_{\mathrm{p}}, \quad E_{\mathrm{D}} = E_{\mathrm{p}} - E_{\mathrm{k}} \tag{16}$$

$$E_{\mathrm{T,c}} = E_{\mathrm{k,c}} + E_{\mathrm{p,c}}, \quad E_{\mathrm{D,c}} = E_{\mathrm{p,c}} - E_{\mathrm{k,c}}. \tag{17}$$

The reason for using E_{T} is: assuming that the flight path angle γ_{fp} and angle of attack α_{aa} are small and that the thrust F_{p} and drag D are aligned, the forces balance is

$$V_{\mathrm{a}}(F_{\mathrm{p}} - D) = g\dot{E}_{\mathrm{T}} \tag{18}$$

that is, the thrust proportionally alters the rate of total energy. Therefore, the thrust is used to control E_{T} via a PID loop with a feedforward term T_{ff}

$$T_{\mathrm{ff}} = T_{\mathrm{D}} + k_{\mathrm{T,ff}} \dot{E}_{\mathrm{T,c}} + k_{\mathrm{T}_\phi} \left(\frac{1}{\cos^2\phi} - 1 \right) \tag{19}$$

$$\delta_{\mathrm{t}} = T_{\mathrm{ff}} + K_{\mathrm{P}_{\mathrm{thr}}} (E_{\mathrm{T,c}} - E_{\mathrm{T}}) + K_{\mathrm{I}_{\mathrm{thr}}} \int_{t_0}^{t} (E_{\mathrm{T,c}}(\tau) - E_{\mathrm{T}}(\tau)) d\tau \tag{20}$$

where T_{D} is the thrust needed to counteract the drag force, $k_{\mathrm{T,ff}}$ controls the feedforward amount and k_{T_ϕ} accounts for the increased drag during aircraft banking.

On the other hand, it is known from physics that the elevator deflection is approximately energy conservative, i.e. it exchanges potential energy for kinetic energy, and vice versa. So, the elevator can be used to control E_{D}. This is done by defining a commanded pitch Θ_{c} (which will be a set point for the low level pitch control in (8)), controlled via another PID loop with feedforward

$$\Theta_{\mathrm{c}} = \frac{1}{V_{\mathrm{a}}} K_{\mathrm{P}_\Theta} (E_{\mathrm{D,c}} - E_{\mathrm{D}}) + \frac{\dot{E}_{\mathrm{D,c}}}{g} + K_{\mathrm{D}_\Theta} (\dot{E}_{\mathrm{D,c}} - \dot{E}_{\mathrm{D}}) + K_{\mathrm{I}_\Theta} \int_{t_0}^{t} (E_{\mathrm{D,c}}(\tau) - E_{\mathrm{D}}(\tau)) d\tau. \tag{21}$$

The overall TECS scheme is shown in Fig. 1(d). Finally, let us mention that although the PID loops in the ArduPilot documentation are in continuous time [25], they are eventually discretized for real implementation.

III. PRELIMINARIES ON MODEL-FREE ADAPTIVE CONTROL

This section is meant to give some preliminaries on model-free adaptive control, so that it is easier to understand how to adopt this method in the ArduPilot architecture. We follow a similar notation as [32]: consider an unknown discrete-time single-input single output (SISO) nonlinear system

$$y(k+1) = f(y(k), y(k-1), \ldots, y(k-n_y), u(k), \ldots, u(k-n_u)) \tag{22}$$

where $u(k) \in \mathbb{R}$ and $y(k) \in \mathbb{R}$ are the control input and system output at time k, n_y, $n_u \in Z_+$ are two unknown orders of output and input, and f is an unknown nonlinear function.

Under some regularity assumptions (the nonlinear dynamics $f(\cdot)$ satisfy Lipschitz continuity and the partial derivatives of $f(\cdot)$ with respect to all variables are continuous [32]), we have that system (22) can be transformed into

$$y(k+1) - y(k) = \varphi_{f,L_y,L_u}^T(k) \Delta H_{L_y,L_u}(k) \tag{23}$$

with $||\varphi_{f,L_y,L_u}(k)|| \leq b$ for any time k, where $H_{L_y,L_u}(k) = [y(k), \ldots, y(k-L_y+1), u(k), \ldots, u(k-L_u+1)]^T \in \mathbb{R}^{L_u+L_y}$ is a vector that includes past input and output data. Denote $\varphi_{f,L_y,L_u}(k) = [\varphi_1(k), \ldots, \varphi_{L_y}(k), \varphi_{L_y+1}(k), \ldots, \varphi_{L_y+L_u}(k)]^T$. In view of (23), the integers L_y ($1 \leq L_y \leq n_y$) and L_u ($1 \leq L_u \leq n_u$) are called output linearization length and input linearization length, and they determine the order of the control law.

Model-free adaptive control considers the following one-step-ahead cost function:

$$J(u(k)) = |y_d(k+1) - y(k+1)|^2 + \lambda |u(k) - u(k-1)|^2 \tag{24}$$

where $\lambda > 0$ is a weighting factor and $y_d(k+1)$ is the desired output signal. The cost function (24) depends on the input to be designed: substituting (23) into (24), and minimizing (24) with respect to $u(k)$ yields the controller

$$u(k) - u(k-1) = \begin{cases} \frac{\varphi_{L_y+1}(k)[\rho_{L_y+1}(y_d(k+1)-y(k)) - \sum_{i=1}^{L_y} \rho \varphi_i(k)(y(k-i+1)-y(k-i))]}{\lambda + |\varphi_{L_y+1}(k)|^2} \\ \quad - \frac{\varphi_{L_y+1}(k) \sum_{i=L_y+2}^{L_y+L_u} \rho \varphi_i(k) \Delta u(k+L_y-i+1)}{\lambda + |\varphi_{L_y+1}(k)|^2}, \qquad L_u \geq 2 \\ \frac{\varphi_{L_y+1}(k)[\rho_{L_y+1}(y_d(k+1)-y(k))}{\lambda + |\varphi_{L_y+1}(k)|^2} \\ \quad \frac{-\sum_{i=1}^{L_y} \rho \varphi_i(k)(y(k-i+1)-y(k-i))]}{\lambda + |\varphi_{L_y+1}(k)|^2}, \qquad L_u = 1 \end{cases} \tag{25}$$

where $\rho \in (0, 1]$ is an auxiliary gain to make the algorithm more flexible.

To derive the controller, a new cost function using the input/output data of the controlled plant is considered

$$\begin{aligned} J(\varphi_{f,L_y,L_u}(k)) = |y(k) - y(k-1) \\ - \varphi^T_{f,L_y,L_u}(k) \Delta H_{L_y,L_u}(k-1)|^2 \\ + \mu ||\varphi_{f,L_y,L_u}(k) - \hat{\varphi}_{f,L_y,L_u}(k-1)||^2 \end{aligned} \tag{26}$$

where $\mu > 0$ is a weighting factor, and $\hat{\varphi}_{f,L_y,L_u} \in \mathbb{R}^{L_y+L_u}$ is the estimate of φ_{f,L_y,L_u}. Minimizing the cost (26) with respect to $\varphi_{f,L_y,L_u}(k)$ gives the following gradient-based estimation

$$\begin{aligned} \hat{\varphi}_{f,L_y,L_u}(k) = \hat{\varphi}_{f,L_y,L_u}(k-1) \\ + \frac{\eta \Delta H_{L_y,L_u}(k-1)(y(k) - y(k-1))}{\mu + ||\Delta H_{L_y,L_u}(k-1)||^2} \\ - \frac{\eta \Delta H_{L_y,L_u}(k-1) \hat{\varphi}^T_{f,L_y,L_u}(k-1) \Delta H_{L_y,L_u}(k-1)}{\mu + ||\Delta H_{L_y,L_u}(k-1)||^2} \end{aligned} \tag{27}$$

$$\begin{aligned} \hat{\varphi}_{f,L_y,L_u}(k) = \hat{\varphi}_{f,L_y,L_u}(1) \text{ if } ||\hat{\varphi}_{f,L_y,L_u}(k)|| \leq \epsilon \\ \text{or} \quad ||\Delta H_{L_y,L_u}(k-1)|| \leq \epsilon \\ \text{or} \quad \text{sgn}(\hat{\varphi}_{f,L_y,L_u}(k)) \neq \text{sgn}(\hat{\varphi}_{f,L_y,L_u}(1)) \end{aligned} \tag{28}$$

where $\eta \in (0, 2]$ represents the update step of the gradient estimation and $\epsilon > 0$ is to avoid division by zero. As a result, the control input $u(k)$ in (25) can be obtained as

$$u(k) - u(k-1) = \begin{cases} \hat{\xi}_{L_y+1}(k)[\rho_{L_y+1}(y_d(k+1) - y(k)) \\ - \sum_{i=1}^{L_y} \rho_i \hat{\varphi}_i(k)(y(k-i+1) - y(k-i)) \\ - \sum_{i=L_y+2}^{L_y+L_u} \rho_i \hat{\varphi}_i(k) \\ (u(k+L_y-i+1) - u(k+L_y-i))], \qquad L_u \geq 2 \\ \hat{\xi}_{L_y+1}(k)[\rho_{L_y+1}(y_d(k+1) - y(k)) \\ - \sum_{i=1}^{L_y} \rho_i \hat{\varphi}_i(k)(y(k-i+1) - y(k-i))], \qquad L_u = 1 \end{cases} \tag{29}$$

TABLE I
List of Low-Level and TECS Gains in ArduPilot

roll		pitch		yaw	
K_{P_ϕ}	0.28	K_{P_θ}	0.36	K_{P_ψ}	0
K_{I_ϕ}	0.045	K_{I_θ}	0.15	K_{I_ψ}	1
K_{D_ϕ}	0.01	K_{D_θ}	0.08	K_{D_ψ}	1.5
Ω_ϕ	2.22	Ω_θ	2.22		

TECS					
T_{D}	45	$K_{\mathrm{P}_{\mathrm{thr}}}$	0.5	$K_{\mathrm{P}\Theta}$	0
$k_{\mathrm{T,ff}}$	0.007	$K_{\mathrm{I}_{\mathrm{thr}}}$	0.1	$K_{\mathrm{I}\Theta}$	0.1
k_{T_ϕ}	10	g	9.81	$K_{\mathrm{D}\Theta}$	0

TABLE II
List of Parameters Used in MFAC and SMC

	MFAC				SMC	
	η	μ	ρ	λ	k	ϵ
roll	1	3.2	0.01	0.32	10.4	0.24
pitch	1	0.01	0.045	0.1	1.9	14.5
slip	0.002	0.4	0.35	1.1	0.05	0.01
TECS1	0.01	6	1	0.1	40	0.003
TECS2	1	17.3	0.01	0.66	2.0	11.3

where $\hat{\xi}_{L_y+1}(k) = \hat{\varphi}_{L_y+1}(k)/(\lambda + |\hat{\varphi}_{L_y+1}(k)|^2) \in \mathbb{R}$. The next section will explain how the control (29) and adaptive laws (27)–(28) are integrated with the ArduPilot architecture.

IV. INTEGRATION OF ARDUPILOT AND MFAC

The numerical values of the gains used in each of the five loops for ArduPilot are in Table I. These gains are in line with those presented in [43], which are the result of gains tuned on Bixler UAV, using the AutoTune procedure of ArduPilot [44].

The ArduPilot architecture is integrated as in Fig. 1(a)–(d) with the MFAC scheme, which only requires to select the order of the regressors and input/output data. We choose the order of the regressors to be $n_y = 2, n_u = 1$, as the resulting control becomes similar to a PID controller with adaptive gains [32]. The other gains (cf., Table II) have been tuned by trial and error, according to the intuitions in [32]: the adaptation step η regulates how fast $\hat{\varphi}$ can be adapted; μ and λ avoid division by zero in the denominators (27) and (29), so they can be selected smaller than some expected average value of $||\Delta H_{L_y,L_u}(k-1)||^2$ and $|\hat{\varphi}_{L_y+1}(k)|^2$; $\rho \in (0, 1]$ is an auxiliary gain to make the controller algorithm more flexible. The input/output data of the MFAC scheme are the following.

1) *Roll loop:* $y = \phi, y_d = \text{LIM}(\Omega(\phi_{\mathrm{c}} - \phi))$, where LIM is the limiter (7), and

$$\delta_{\mathrm{a}}(k) = u_{\mathrm{PID},\delta_{\mathrm{a}}}(k) + \sigma_\phi \, u_{\mathrm{MFAC},\delta_{\mathrm{a}}}(k)$$

where $u_{\mathrm{PID},\delta_{\mathrm{a}}}$ is calculated from (6) and $u_{\mathrm{MFAC},\delta_{\mathrm{a}}}$ is calculated from (29) with the aforementioned output data.

2) *Pitch loop:* $y = \theta$, $y_d = \text{LIM}(\Omega(\theta_{\mathrm{c}} - \theta)) + \theta_{\mathrm{bank}}$, where LIM is the limiter (9), θ_{bank} is the roll compensation (10), and

$$\delta_{\mathrm{e}}(k) = u_{\mathrm{PID},\delta_{\mathrm{e}}}(k) + \sigma_\theta \, u_{\mathrm{MFAC},\delta_{\mathrm{e}}}(k)$$

where u_{PID,δ_e} is calculated from (8) and u_{MFAC,δ_e} is calculated from (29) with the aforementioned output data.

3) *Side-slip loop:* $y = a_y$, $y_d = H(s)(\dot{\psi} - \dot{\psi}_{\text{turn}})$, where $\dot{\psi}_{\text{turn}}$ is the turn coordination (12), $H(s)$ is the high-pass filter (13), and

$$\delta_r(k) = u_{\text{PID},\delta_r}(k) + \sigma_\psi \; u_{\text{MFAC},\delta_r}(k)$$

where u_{PID,δ_r} is calculated from (11) and u_{MFAC,δ_r} is calculated from (29) with the aforementioned output data.

4) *TECS1 loop (throttle demand):* $y = E_{\text{T}}$, $y_d = E_{\text{T,c}}$, and

$$\delta_t(k) = u_{\text{PID},\delta_t}(k) + \sigma_{\text{thr}} \; u_{\text{MFAC},\delta_t}(k)$$

where u_{PID,δ_t} is calculated from (20) and u_{MFAC,δ_t} is calculated from (29) with the aforementioned output data.

5) *TECS2 loop (pitch demand):* $y = E_{\text{D}}$, $y_d = E_{\text{D,c}}$, and

$$\Theta_c(k) = u_{\text{PID},\Theta_c}(k) + \sigma_\Theta \; u_{\text{MFAC},\Theta_c}(k)$$

where u_{PID,θ_c} is calculated from (21) and u_{MFAC,θ_c} is calculated from (29) with the aforementioned output data.

The gains $\sigma_\phi \in \{0, 1\}$, $\sigma_\theta \in \{0, 1\}$, $\sigma_\psi \in \{0, 1\}$, $\sigma_{\text{thr}} \in \{0, 1\}$, $\sigma_\Theta \in \{0, 1\}$ are binary gains used to activate or deactivate the adaptation module in the corresponding loop. The schemes of the augmented control system are given in the diagrams of Fig. 1(a)–(d). The control system is modular and the model-free adaptive control methodology is integrated in both the low-level control (roll/pitch/yaw) and the TECS.

V. SOFTWARE-IN-THE-LOOP EXPERIMENTS

The tests are conducted for a fixed-wing UAV that must follow a line and then orbit around a point. During the flight, done at approximately 15 m/s airspeed, the UAV faces a wind of 4 m/s. The wind is also affected by Dryden turbulence, in line with [27, Sec. 4.4]. The following three scenarios are considered.

1) mass = 1kg;
2) mass = 1.5kg;
3) mass = 0.5kg.

These scenarios allow to evaluate how different controllers behave in the presence of uncertain payloads. To make the tests realistic, the ArduPilot functionalities have been emulated in MATLAB according to the ArduPilot documentation and code, which allows to perform software-in-the-loop tests. More details on this software-in-the-loop platform, developed and maintained by some of the authors, are in [42] and[43]. The simulation environment comprises sensor measurement noises and control deflection limits (aileron: $\pm 30^\circ$, elevator: $\pm 15^\circ$, rudder: $\pm 25^\circ$).

A. Comparisons and Results

The original ArduPilot is used as baseline. To evaluate the effect of augmenting different loops of the original ArduPilot architecture, many combinations are examined, e.g. augmenting one loop, augmenting two loops, until all five loops are augmented. In addition, to compare with an alternative method, we consider a methodology inspired by robust sliding mode control (SMC), motivated by recent works [45], [46] for fixed-wing UAVs. The robust SMC can be written as

$$\delta_a(k) = u_{\text{PID},\delta_a}(k) + k_a \,\text{sat}(s_a(k)/\epsilon_a)$$

$$\delta_e(k) = u_{\text{PID},\delta_e}(k) + k_e \,\text{sat}(s_e(k)/\epsilon_e)$$

$$\delta_r(k) = u_{\text{PID},\delta_t}(k) + k_t \,\text{sat}(s_t(k)/\epsilon_t)$$

$$\delta_t(k) = u_{\text{PID},\delta_r}(k) + k_r \,\text{sat}(s_r(k)/\epsilon_r)$$

$$\Theta_c(k) = u_{\text{PID},\Theta_c}(k) + k_c \,\text{sat}(s_c(k)/\epsilon_c)$$

where u_{PID,δ_a}, u_{PID,δ_e}, u_{PID,δ_t}, u_{PID,δ_r}, u_{PID,Θ_c} are the contributions of the original ArduPilot, k_a, k_e, k_t, k_r, k_c are the switching gains of sliding mode control [45], [46], s_a, s_e, s_t, s_r, s_c are the sliding surfaces, and ϵ_a, ϵ_e, ϵ_t, ϵ_r, ϵ_c define the size of the saturation. In our tests, $s_a = u_{\text{PID},\delta_a}$, $s_e = u_{\text{PID},\delta_e}$, $s_t = u_{\text{PID},\delta_t}$, $s_r = u_{\text{PID},\delta_r}$, $s_c = u_{\text{PID},\Theta_c}$. The numerical values of SMC gains, tuned by trial and error, are in Table II.

The comparisons are reported in Tables III (tracking and input norm for robust SMC) and IV (tracking and input norm for proposed adaptation). The tracking error contribution is

$$\begin{aligned}\text{cost}_e = \frac{1}{T_{\text{fin}}} \sum_{k=1}^{T_{\text{fin}}} \big[&(\text{LIM}(\Omega(\phi_c(k) - \phi(k))) - \phi(k))^2 \\ &+ (\Omega(\theta_c(k) - \theta(k))) - \theta(k))^2 \\ &+ (H(s)(\dot{\psi}(k) - \dot{\psi}_{\text{turn}}(k)) - a_y(k))^2 \\ &+ (E_{\text{T,c}}(k) - E_{\text{T}}(k))^2 + (E_{\text{D,c}}(k) - E_{\text{D}}(k))^2\big]\end{aligned} \tag{30}$$

and the control input contribution (control effort) is

$$\text{cost}_u = \frac{1}{T_{\text{fin}}} \sum_{k=1}^{T_{\text{fin}}} \big[\delta_a^2(k) + \delta_e^2(k) + \delta_r^2(k) + \delta_t^2(k) + \Theta_c^2(k)\big] \tag{31}$$

that is, each experiment with robust SMC or proposed adaptation accounts for the contribution of all loops, even when not all loops in Tables III and IV are augmented. The percentage variations are calculated with respect to the original ArduPilot. The following can be seen from Tables III and IV.

1) The model-free adaptive control augmentation is always beneficial for tracking (the tracking cost is always reduced in Table IV).

TABLE III
Tracking and Control Costs When Augmenting Different Loops of the Original ArduPilot Architecture With Robust SMC

Add SMC	mass $= 1kg$		mass $= 1.5kg$		mass $= 0.5kg$	
	Tracking	Control	Tracking	Control	Tracking	Control
original ArduPilot (no SMC)	43.62	4.49	52.99	5.47	36.40	3.76
roll	43.01 (-1.4%)	4.49 (+0.1%)	52.25 (-1.4%)	5.47 (-0.1%)	37.27 (+2.4%)	3.79 (+0.7%)
pitch	43.76 (+0.3%)	4.52 (+0.6%)	49.08 (-7.4%)	5.49 (+0.4%)	42.17 (+15.8%)	3.88 (+3.3%)
slip	43.67 (+0.1%)	4.51 (+0.4%)	53.06 (+0.1%)	5.49 (+0.3%)	36.49 (+0.3%)	3.78 (+0.6%)
tecs1	43.62 (0.0%)	4.49 (0.0%)	52.99 (0.0%)	5.47 (0.0%)	36.40 (0.0%)	3.76 (0.0%)
tecs2	14.95 (-65.7%)	4.64 (+3.3%)	24.96 (-52.9%)	5.82 (+6.3%)	22.17 (-39.1%)	4.36 (+16.0%)
roll+pitch	43.26 (-0.8%)	4.52 (+0.8%)	48.48 (-8.5%)	5.49 (+0.4%)	45.09 (+23.9%)	4.01 (+6.7%)
pitch+slip	43.82 (+0.5%)	4.54 (+1.0%)	49.15 (-7.3%)	5.51 (+0.8%)	42.46 (+16.7%)	3.92 (+4.3%)
pitch+tecs1	43.76 (+0.3%)	4.52 (+0.6%)	49.07 (-7.4%)	5.49 (+0.4%)	42.17 (+15.8%)	3.88 (+3.3%)
tecs1+tecs2	14.94 (-65.7%)	4.64 (+3.3%)	24.96 (-52.9%)	5.82 (+6.3%)	22.17 (-39.1%)	4.36 (+16.0%)
roll+pitch+slip	43.30 (-0.7%)	4.54 (+1.2%)	48.55 (-8.4%)	5.51 (+0.7%)	45.18 (+24.1%)	4.04 (+7.5%)
roll+slip+tecs1	43.05 (-1.3%)	4.51 (+0.5%)	52.30 (-1.3%)	5.49 (+0.3%)	37.42 (+2.8%)	3.81 (+1.4%)
roll+pitch+slip+tecs1	43.30 (-0.8%)	4.54 (+1.2%)	48.55 (-8.4%)	5.51 (+0.7%)	45.19 (+24.1%)	4.04 (+7.5%)
roll+pitch+slip+tecs2	13.70 (-68.6%)	4.58 (+2.1%)	17.85 (-66.3%)	5.73 (+4.7%)	19.75 (-45.8%)	4.05 (+7.7%)
roll+pitch+tecs1+tecs2	13.68 (-68.6%)	4.56 (+1.7%)	17.81 (-66.4%)	5.71 (+4.4%)	19.39 (-46.7%)	3.99 (+6.2%)
pitch+slip+tecs1+tecs2	14.09 (-67.7%)	4.58 (+1.9%)	18.35 (-65.4%)	5.72 (+4.6%)	19.50 (-46.4%)	4.07 (+8.3%)
all SMC	13.69 (-68.6%)	4.58 (+2.1%)	17.85 (-66.3%)	5.73 (+4.7%)	19.75 (-45.8%)	4.05 (+7.7%)

The percentage variations are calculated with respect to the original ArduPilot.

TABLE IV
Tracking and Control Costs When Augmenting Different Loops of the Original ArduPilot Architecture With Adaptation

Add MFAC	mass $= 1kg$		mass $= 1.5kg$		mass $= 0.5kg$	
	Tracking	Control	Tracking	Control	Tracking	Control
original ArduPilot (no MFAC)	43.62	4.49	52.99	5.47	36.40	3.76
roll	41.11 (-5.8%)	4.50 (+0.3%)	50.06 (-5.5%)	5.48 (+0.2%)	33.14 (-9.0%)	3.69 (-1.8%)
pitch	24.52 (-43.8%)	4.37 (-2.6%)	33.50 (-36.8%)	5.42 (-1.0%)	17.85 (-51.0%)	3.63 (-3.5%)
slip	41.23 (-5.5%)	3.81 (-15.1%)	50.56 (-4.6%)	4.78 (-12.7%)	32.82 (-9.9%)	3.09 (-17.8%)
tecs1	41.59 (-4.7%)	4.47 (-0.4%)	51.00 (-3.8%)	5.45 (-0.4%)	34.59 (-5.0%)	3.75 (-0.1%)
tecs2	28.68 (-34.3%)	4.68 (+4.3%)	34.24 (+35.4%)	5.87 (+7.4%)	26.28 (-27.8%)	3.99 (+6.1%)
roll+pitch	22.00 (-49.6%)	4.38 (-2.4%)	31.10 (-41.3%)	5.44 (-0.6%)	13.44 (-63.1%)	3.50 (-7.0%)
pitch+slip	22.16 (-49.2%)	3.69 (-17.7%)	31.41 (-40.7%)	4.75 (-13.3%)	13.72 (-62.3%)	2.90 (-23.0%)
pitch+tecs1	22.25 (-49.0%)	4.36 (-3.0%)	31.80 (-40.0%)	5.39 (-1.5%)	16.01 (-56.0%)	3.63 (-3.5%)
tecs1+tecs2	25.73 (-41.0%)	4.68 (+4.2%)	32.19 (-39.3%)	5.87 (+7.2%)	23.99 (-34.1%)	4.00 (+6.6%)
roll+pitch+slip	20.98 (-51.9%)	3.73 (-17.0%)	30.69 (-42.1%)	4.83 (-11.8%)	12.34 (-66.1%)	2.92 (-22.3%)
roll+slip+tecs1	38.16 (-12.5%)	3.83 (-14.7%)	47.98 (-9.5%)	4.81 (-12.0%)	30.18 (-17.1%)	3.11 (-17.3%)
roll+pitch+slip+tecs1	19.06 (-56.3%)	3.72 (-17.2%)	28.59 (-46.0%)	4.79 (-12.5%)	10.04 (-72.4%)	2.91 (-22.5%)
roll+pitch+slip+tecs2	11.40 (-73.9%)	3.79 (-15.6%)	19.95 (-62.4%)	5.10 (-6.8%)	9.50 (-73.9%)	2.92 (-22.3%)
roll+pitch+tecs1+tecs2	9.63 (-77.9%)	4.41 (-1.8%)	17.21 (-67.5%)	5.70 (+4.1%)	9.62 (-73.6%)	3.52 (-6.3%)
pitch+slip+tecs1+tecs2	9.93 (-77.2%)	3.76 (-16.3%)	19.46 (-63.3%)	5.05 (-7.7%)	8.18 (-77.5%)	2.89 (-23.2%)
all MFAC	8.48 (-80.6%)	3.78 (-15.9%)	15.70 (-70.4%)	5.06 (-7.6%)	7.12 (-80.4%)	2.91 (-22.7%)

The percentage variations are calculated with respect to the original ArduPilot.

2) The same does *not* hold for the robust SMC in Table III, where sometimes the tracking cost is larger than ArduPilot.
3) The robust SMC usually increases the control effort (increasing control costs in Table III).
4) On the contrary, the model-free adaptive control augmentation generally reduces the control effort (smaller control costs in Table IV).

Decreasing the control effort while improving tracking is a remarkable feature of the proposed scheme. The best improvement is obtained when all five loops of ArduPilot are augmented with adaptation. To the best of our knowledge, it is the first time that model-free adaptive control is integrated system-wide in a complete ArduPilot-based autopilot architecture. Table IV reveals that the improvements are consistent for all masses: the full augmentation leads to tracking improvements in the range 70.4%–80.4% and control reduction in the range 7.6%–22.7%. The robust SMC can improve tracking by 45.8%–68.6% but it requires increased control effort in the range 2.1%–7.7%.

To understand the reasons for such improvements, we report in Fig. 2 the paths under the original ArduPilot, the robust SMC, and the proposed augmentation. The UAV should first follow a desired line (in order to do this, the UAV must circumnavigate the line). Due to the wind, the original ArduPilot deviates from the desired path and slowly converges to the line. When the UAV orbits around a desired point, the zoom in Fig. 2 show that the wind makes the original ArduPilot deviate almost one meter from the desired orbit. The proposed autopilot is less affected by the wind for all masses of the UAV, e.g., it deviates only 30 cm

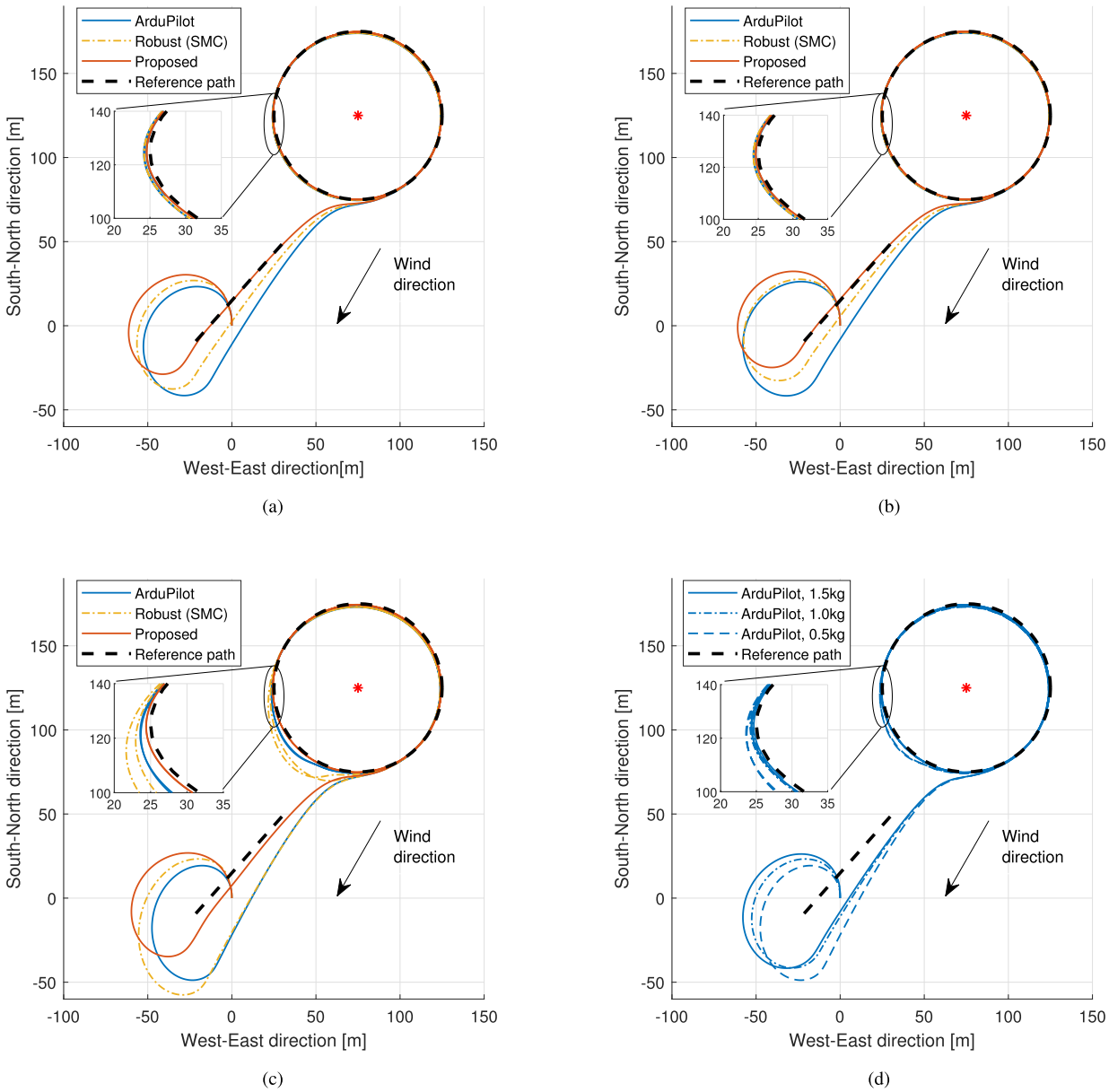


Fig. 2. Paths on the $(x-y)$ plane for different masses, with original PID ArduPilot and with proposed augmentation. Note that the UAV is less affected by wind when mass=1.5 kg and more affected when mass=0.5 kg. (a) Path for mass=1kg. (b) Path for mass=1.5kg. (c) Path for mass=0.5kg. (d) Comparison of ArduPilot paths.

from the orbit. The robust SMC works better than ArduPilot (but worse than the proposed adaptive augmentation) for mass=1 kg and mass=1.5 kg. However, for mass=0.5 kg, a substantial degradation can be noticed, especially during the orbit phase. Note that the UAV is less affected by wind when mass=1.5 kg and more affected when mass=0.5 kg, which is consistent with intuition.

To further understand the improvements of the proposed approach, we also report the tracking and control results for all loops. Fig. 3 report the tracking errors (with original ArduPilot, with robust SMC and with proposed augmentation), while Fig. 4 reports the corresponding inputs (with original ArduPilot, with robust SMC and with proposed augmentation). It is clear to see that the errors of the augmented architecture are much smaller than the errors of the original architecture. The fact that the errors are kept small for any mass certifies the consistency of the augmentation method.

B. Mass Change and Time-Varying Wind

To further test the proposed approach, we propose a scenario with changing mass: the UAV mass is initially 1.5 kg and it suddenly drops to 0.5 kg. This scenario can simulate the UAV suddenly releasing or losing its payload. Clearly, the autopilot should compensate for this sudden

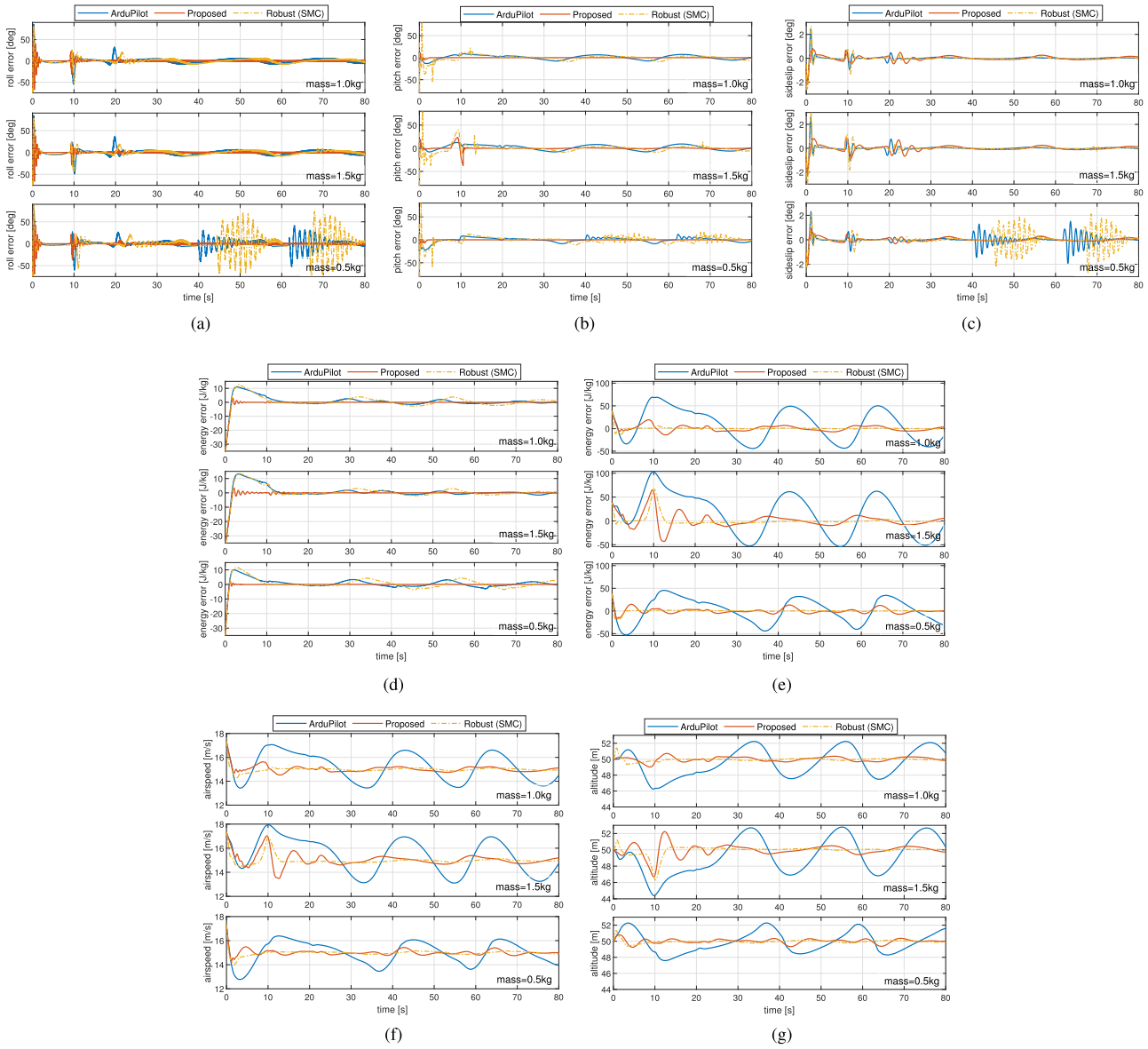


Fig. 3. Tracking errors for all five loops (roll, pitch, yaw, TECS1, and TECS2) for different masses: 1 kg (top), 1.5 kg (middle), 0.5 kg (bottom). Airspeed and altitude are also reported. The original ArduPilot and the proposed augmentation have solid lines, the robust SMC has dash-dot line. (a) Roll errors. (b) Pitch errors. (c) Side-slip errors. (d) Energy (TECS1) errors. (e) Energy (TECS2) errors. (f) Airspeed. (g) Altitude.

change. At the same time, a time varying wind is created, i.e., the wind of 4 m/s is sinusoidally perturbed in magnitude (perturbation of 1 m/s) and in direction (perturbation of 90°). The resulting paths are reported in Fig. 5 in the (x–y) plane: again, it can be seen that the proposed adaptive autopilot improves over the original ArduPilot and the robust SMC. One can notice that the mass change causes a large perturbation in both ArduPilot and the robust SMC autopilot, whereas the proposed adaptive autopilot is almost unaffected by the mass change. Full details of inputs and tracking errors are not reported for lack of space, but Table V provides the tracking and control costs for the different autopilot architectures: the full augmentation leads to tracking improvement of 79.5% and control reduction of 28.8%. The robust SMC can improve tracking of 38.2% but it requires increased control effort of 6.1%.

TABLE V
Tracking and Control Costs With Mass Change and Time-Varying Wind for Different Autopilots

Autopilot	mass $= 1.5 \rightarrow 0.5$kg	
	Tracking	Control
Original ArduPilot	76.41	7.04
Robust SMC autopilot	47.23 (-38.2%)	7.47 (+6.1%)
Proposed adaptive autopilot	15.66 (-79.5%)	5.02 (-28.8%)

The percentage variations are calculated with respect to the original ArduPilot.

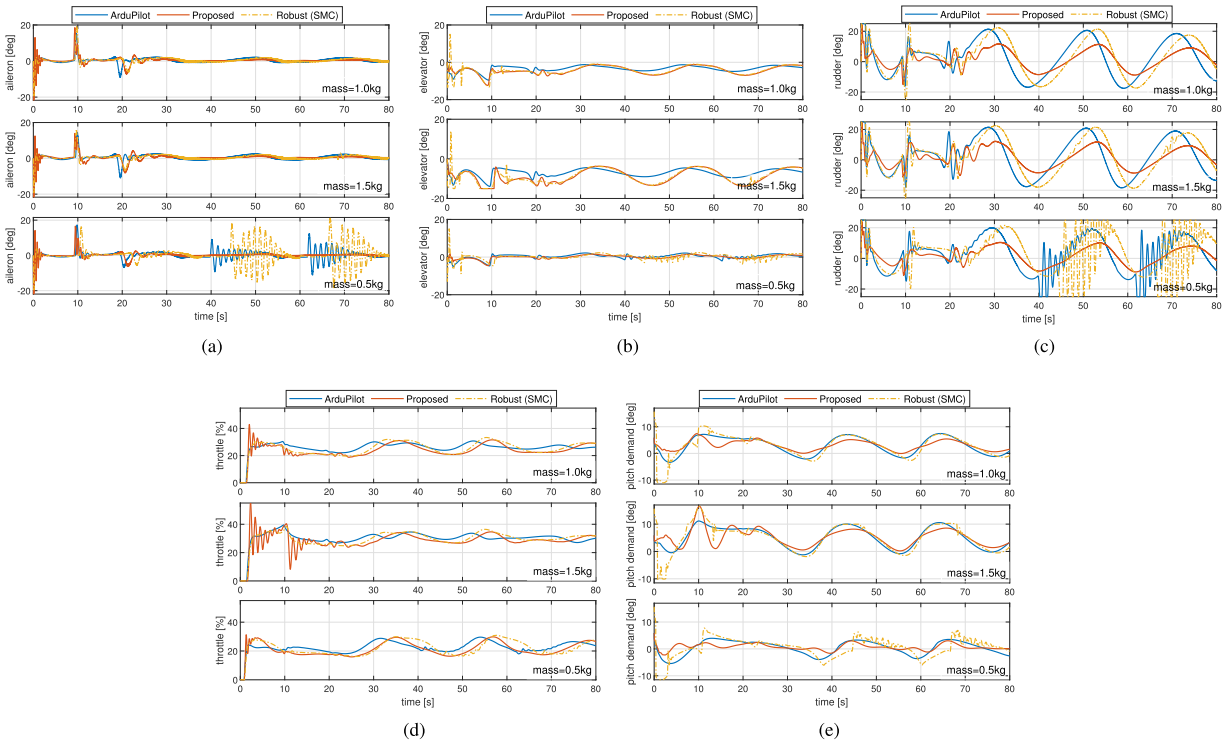


Fig. 4. Control inputs for all five loops (roll, pitch, yaw, TECS1, and TECS2) for different masses: 1 kg (top), 1.5 kg (middle), 0.5 kg (bottom). The original ArduPilot and the proposed augmentation have solid lines, the robust SMC has dash-dot line. (a) Roll inputs (aileron). (b) Pitch inputs (elevator). (c) Side-slip inputs (rudder). (d) TECS1 inputs (throttle). (e) TECS2 inputs (pitch demand).

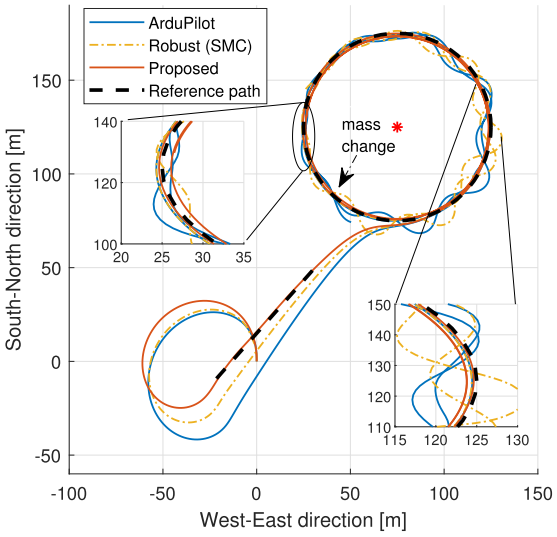


Fig. 5. Paths on the $(x-y)$ plane for mass change $1.5 \rightarrow 0.5$ kg and time-varying wind, with original PID ArduPilot, with robust SMC and with proposed augmentation. One can notice that the mass change causes a large perturbation in both ArduPilot and the robust SMC autopilot.

VI. CONCLUSION

This article has presented an adaptive augmentation method for ArduPilot-based autopilots of fixed-wing UAVs. This augmentation strategy was adopted for both attitude and total energy control loops of ArduPilot, an open-source software suite developed and maintained by a large UAV community. The augmented architecture was tested in a software-in-the-loop UAV platform in the presence of several uncertainties (represented by different payloads of the UAV, low-level unmodeled dynamics, and time-varying wind). The performance was measured in terms of the tracking errors and control effort of the attitude and total energy control loops. Extensive comparative experiments with the original ArduPilot, with the proposed augmentation, and with an alternative robust autopilot have shown that the augmentation method can significantly improve the performance of the UAV consistently for all payload and wind conditions.

Interesting future research directions include considering practical aspects of the UAV from an analytic point of view, such as the presence of saturation (which might require an antiwind up design) and the presence of measurement noise (which might require a model-free observer). Eventually, testing these algorithm in real-life is also of interest.

APPENDIX
UAV DYNAMICS

This appendix recalls basic notions of UAV dynamics and gives some parameters of the UAV used in this study. More complete discussions of UAV dynamics can be found in [27, Ch. 3] and a complete list of parameters is in [43]. Fig. 6 illustrates the axes of motion of the UAV. The relationship between translational velocity and inertial position

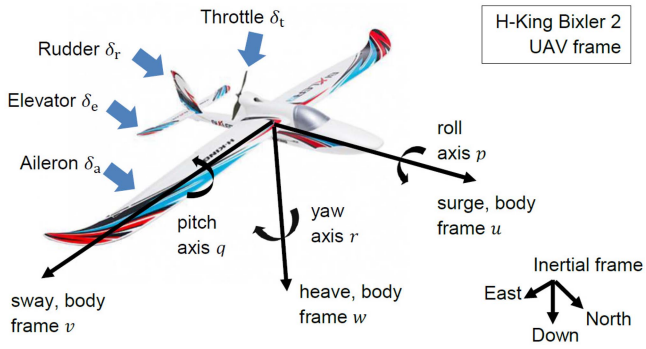


Fig. 6. Axes of motion for fixed-wing UAV.

of the UAV requires a differentiation and rotation

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\psi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\psi s_\theta s_\psi - s_\psi c_\psi & c_\phi c_\theta \end{bmatrix}^T \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

with the short notation $c_\psi = \cos\phi$, $s_\phi = \sin\phi$, and where u, v, w are the inertial velocity components projected onto the body frame. The body-frame angular rates can be expressed in terms of the Euler angles ϕ, θ, ψ

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\,\tan\theta & \cos\phi\,\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\psi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

where p, q, r are roll/pitch/yaw rate measured along the corresponding axes. For translational motion, it holds that

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$$

where f_x, f_y, f_z are the externally applied forces defined in terms of body frame.

The rotational dynamics equations are

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6 \left(p^2 - r^2\right) \\ \Gamma_7 pq - \Gamma_1 qr \end{bmatrix} + \begin{bmatrix} \Gamma_3 \mathcal{L} + \Gamma_4 \mathcal{N} \\ \frac{1}{J_y}\mathcal{M} \\ \Gamma_4 \mathcal{L} + \Gamma_8 \mathcal{N} \end{bmatrix}$$

where

$$\Gamma_1 = \frac{J_{xz}\left(J_x - J_y + J_z\right)}{\Gamma} \qquad \Gamma_2 = \frac{J_z\left(J_z - J_y\right) + J_{xz}^2}{\Gamma}$$
$$\Gamma_3 = \frac{J_z}{\Gamma} \qquad \Gamma_4 = \frac{J_{xz}}{\Gamma}$$
$$\Gamma_5 = \frac{J_z - J_x}{J_y} \qquad \Gamma_6 = \frac{J_{xy}}{J_y}$$
$$\Gamma_7 = \frac{\left(J_x - J_y\right)J_x + J_{xy}^2}{\Gamma} \qquad \Gamma_8 = \frac{J_x}{\Gamma}$$

$\Gamma = J_x J_z - J_{xz}^2$, with the inertia tensor defined as

$$\mathbf{J} = \begin{bmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{bmatrix}.$$

TABLE VI
Physical Parameter Values

Parameter	Value	Unit	Parameter	Value	Unit
m	1.0	kg			
J_x	0.020	$\mathrm{kg \cdot m^2}$	J_z	0.053	$\mathrm{kg \cdot m^2}$
J_y	0.026	$\mathrm{kg \cdot m^2}$	J_{xz}	0	$\mathrm{kg \cdot m^2}$
S_p	0.031	$\mathrm{m^2}$	R_prop	0.1	m
b_wing	1.31	m	C_prop	0.12	-
c_wing	0.175	m	k_mot	11.39	rad/s
S_wing	0.228	$\mathrm{m^2}$	q_mot	239	rad/s
$\delta_\mathrm{a}^\mathrm{min}$,$\delta_\mathrm{a}^\mathrm{max}$,	± 0.52	rad	$\delta_\mathrm{e}^\mathrm{min}$,$\delta_\mathrm{e}^\mathrm{max}$	± 0.26	rad
$\delta_\mathrm{r}^\mathrm{min}$,$\delta_\mathrm{r}^\mathrm{max}$	± 0.44	rad	$\delta_\mathrm{t}^\mathrm{min}$,$\delta_\mathrm{t}^\mathrm{max}$	[0,100]	%

Note that it is considered that $J_{xy} = J_{yz} = 0$ since most aircraft are symmetric around these planes. The external forces and moments are dependent on some states and input of the UAV. More specifically

$$F_\mathrm{lift} = \frac{1}{2}\rho_\mathrm{air} V_\mathrm{a}^2 S_\mathrm{wing} C_\mathrm{L}\left(\alpha_\mathrm{aa}, q, \delta_\mathrm{e}\right)$$
$$F_\mathrm{drag} = \frac{1}{2}\rho_\mathrm{air} V_\mathrm{a}^2 S_\mathrm{wing} C_\mathrm{D}\left(\alpha_\mathrm{aa}, q, \delta_\mathrm{e}\right)$$
$$\mathcal{M} = \frac{1}{2}\rho_\mathrm{air} V_\mathrm{a}^2 S_\mathrm{wing} c_\mathrm{wing} C_\mathrm{m}\left(\alpha_\mathrm{aa}, q, \delta_\mathrm{e}\right)$$

where ρ_air is the air density, S_wing the planform area of the single wings, c_wing is the main chord of the wing. The nonlinear functions C_L, C_D, C_m are typically approximated via Taylor expansion [27, Ch. 3]. Note that F_lift and F_drag are expressed in the stability frame—to bring them onto the body frame, the following rotation is performed:

$$\begin{bmatrix} f_x \\ 0 \\ f_z \end{bmatrix} = \begin{bmatrix} \cos\beta_\mathrm{sa}\cos\alpha_\mathrm{aa} & \sin\beta_\mathrm{sa}\cos\beta_\mathrm{sa}\sin\alpha_\mathrm{aa} \\ -\sin\beta_\mathrm{sa}\cos\alpha_\mathrm{aa} & \cos\beta_\mathrm{sa} - \sin\beta_\mathrm{sa}\sin\alpha_\mathrm{aa} \\ -\sin\alpha_\mathrm{aa} & 0 \end{bmatrix}^T \begin{bmatrix} \cos\alpha_\mathrm{aa} \\ -F_\mathrm{lift} \\ 0 \\ -F_\mathrm{drag} \end{bmatrix}$$

where α_aa, β_sa are the angle of attack and side slip angle, respectively. Note that the longitudinal force is also affected by the thrust, for which the following equation holds:

$$F_p = S_p C_\mathrm{prop}\left(P_\mathrm{out} - P_\mathrm{in}\right)$$
$$= \frac{1}{2}\rho_\mathrm{air} S_\mathrm{p} C_\mathrm{prop}\left[\left(R_\mathrm{prop}(k_\mathrm{mot}\delta_\mathrm{t} + q_\mathrm{mot})\right)^2 - V_a^2\right]$$

where R_prop is the propeller radius, C_prop is a nondimensional coefficient representing rotor thrust efficiency, and k_mot, q_mot are motor constants [43].

The lateral aerodynamics directly influences the lateral force, as well as the rolling and yawing moments

$$f_y = \frac{1}{2}\rho_\mathrm{air} V_\mathrm{a}^2 S_\mathrm{wing} C_\mathrm{Y}\left(\beta_\mathrm{sa}, p, r, \delta_\mathrm{a}, \delta_\mathrm{r}\right)$$
$$\mathcal{L} = \frac{1}{2}\rho_\mathrm{air} V_\mathrm{a}^2 S_\mathrm{wing} b_\mathrm{wing} C_\mathrm{l}\left(\beta_\mathrm{sa}, p, r, \delta_\mathrm{a}, \delta_\mathrm{r}\right)$$
$$\mathcal{N} = \frac{1}{2}\rho_\mathrm{air} V_\mathrm{a}^2 S_\mathrm{wing} b_\mathrm{wing} C_\mathrm{n}\left(\beta_\mathrm{sa}, p, r, \delta_\mathrm{a}, \delta_\mathrm{r}\right)$$

where b_wing is the wingspan, and C_Y, C_l, C_n are nonlinear functions typically approximated via Taylor expansion [27,

Ch. 3]. The numerical values of the parameters are in Table VI. A more complete list can be found in [43].

REFERENCES

[1] F. Kara and M. U. Salamci
Model reference adaptive sliding surface design for nonlinear systems
IEEE Trans. Ind. Appl., vol. 54, no. 1, pp. 611–624, Jan./Feb. 2018.

[2] J.-P. Cai, L. Xing, M. Zhang, and L. Shen
Adaptive neural network control for missile systems with unknown hysteresis input
IEEE Access, vol. 5, pp. 15839–15847, 2017.

[3] C. Xun and L. Yongshan
Design and analysis of autopilot based on adaptive control
In *Proc. Int. Conf. Mechatronic Sci., Electr. Eng. Comput.*, 2013, pp. 2918–2921.

[4] Y. Zhang
Adaptive control at launching using three loop autopilot
In *Proc. 2nd Int. Conf. Mechanic Autom. Control Eng.*, 2011, pp. 5535–5538.

[5] S. N. Tiwari, P. N. Dwivedi, A. Bhattacharya, and R. Padhi
L1 adaptive dynamic inversion missile autopilot with pi structure and adaptive sampling
In *Proc. Indian Control Conf.*, 2016, pp. 54–59.

[6] J. Du, C. Guo, S. Yu, and Y. Zhao
Adaptive autopilot design of time-varying uncertain ships with completely unknown control coefficient
IEEE J. Ocean. Eng., vol. 32, no. 2, pp. 346–352, Apr. 2007.

[7] Y. Liu and Q. Zhu
Adaptive fuzzy control for ship autopilot with course constraint based on event-triggered mechanism
In *Proc. 7th Int. Conf. Inf., Cybern., Comput. Social Syst.*, 2020, pp. 175–179.

[8] H. Sun, F. Yu, F. He, and J. Sheng
A novel adaptive fuzzy autopilot design based on DSC
In *Proc. Int. Conf. Elect. Control Eng.*, 2011, pp. 3300–3304.

[9] J. Hill and Q. Lam
Using an adaptive autopilot for flight control system performance improvement
In *Proc. 34th IEEE Conf. Decis. Control*, 1995, pp. 3759–3762.

[10] S. Rao and D. Ghose
Sliding mode control-based autopilots for leaderless consensus of unmanned aerial vehicles
IEEE Trans. Control Syst. Technol., vol. 22, no. 5, pp. 1964–1972, Sep. 2014.

[11] J. Wang and M. Xin
Integrated optimal formation control of multiple unmanned aerial vehicles
IEEE Trans. Control Syst. Technol., vol. 21, no. 5, pp. 1731–1744, Sep. 2013.

[12] H. Xie, G. Fink, A. F. Lynch, and M. Jagersand
Adaptive visual servoing of UAVs using a virtual camera
IEEE Trans. Aerosp. Electron. Syst., vol. 52, no. 5, pp. 2529–2538, Oct. 2016.

[13] A. Bosso, C. Conficoni, D. Raggini, and A. Tilli
A computational-effective field-oriented control strategy for accurate and efficient electric propulsion of unmanned aerial vehicles
IEEE/ASME Trans. Mechatronics, vol. 26, no. 3, pp. 1501–1511, Jun. 2021.

[14] B. Zhou, H. Satyavada, and S. Baldi
Adaptive path following for unmanned aerial vehicles in time-varying unknown wind environments
In *Proc. Amer. Control Conf.*, 2017, pp. 1127–1132.

[15] F. Santoso, M. A. Garratt, and S. G. Anavatti
State-of-the-art intelligent flight control systems in unmanned aerial vehicles
IEEE Trans. Autom. Sci. Eng., vol. 15, no. 2, pp. 613–627, Apr. 2018.

[16] A. L'Afflitto, R. B. Anderson, and K. Mohammadi
An introduction to nonlinear robust control for unmanned quadrotor aircraft: How to design control algorithms for quadrotors using sliding mode control and adaptive control techniques [focus on education]
IEEE Control Syst. Mag., vol. 38, no. 3, pp. 102–121, Jun. 2018.

[17] S. Abdelmoeti and R. Carloni
Robust control of UAVs using the parameter space approach
In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 5632–5637.

[18] A. Al-Mahturi, F. Santoso, M. A. Garratt, and S. G. Anavatti
A robust self-adaptive interval type-2 TS fuzzy logic for controlling multi-input-multi-output nonlinear uncertain dynamical systems
IEEE Trans. Syst., Man, Cybern. Syst., vol. 52, no. 1, pp. 655–666, Jan. 2022.

[19] D. Invernizzi, M. Lovera, and L. Zaccarian
Dynamic attitude planning for trajectory tracking in thrust-vectoring UAVs
IEEE Trans. Autom. Control, vol. 65, no. 1, pp. 453–460, Jan. 2020.

[20] D. Invernizzi, M. Lovera, and L. Zaccarian
Integral ISS-based cascade stabilization for vectored-thrust UAVs
IEEE Contr. Syst. Lett., vol. 4, no. 1, pp. 43–48, Jan. 2020.

[21] Y. Wei, H. Xu, and Y. Xue
Adaptive neural networks-based dynamic inversion applied to reconfigurable flight control and envelope protection under icing conditions
IEEE Access, vol. 8, pp. 11577–11594, 2020.

[22] S. Farì, X. Wang, S. Roy, and S. Baldi
Addressing unmodeled path-following dynamics via adaptive vector field: A UAV test case
IEEE Trans. Aerosp. Electron. Syst., vol. 56, no. 2, pp. 1613–1622, Apr. 2020.

[23] A. B. Farjadian, B. Thomsen, A. M. Annaswamy, and D. D. Woods
Resilient flight control: An architecture for human supervision of automation
IEEE Trans. Control Syst. Technol., vol. 29, no. 1, pp. 29–42, Jan. 2021.

[24] S. Yang, K. Li, and J. Shi
Design and simulation of the longitudinal autopilot of UAV based on self-adaptive fuzzy PID control
In *Proc. Int. Conf. Comput. Intell. Secur.*, 2009, pp. 634–638.

[25] Open source for ardupilot open source autopilot
[Online]. Available: https://github.com/ArduPilot/ardupilot.

[26] Open source for drones-px4 open source autopilot
[Online]. Available: https://px4.io/

[27] R. W. Beard and T. W. Mclain
Small Unmanned Aircraft Theory and Practice. Princeton, NJ, USA:Princeton Univ. Press, 2012.

[28] B. L. Stevens, F. L. Lewis, and E. N. Johnson
Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems. Hoboken, NJ, USA:Wiley, 2015.

[29] H. Lee, S. Snyder, and N. Hovakimyan
$\mathcal{L}_1$ adaptive output feedback augmentation for missile systems
IEEE Trans. Aerosp. Electron. Syst., vol. 54, no. 2, pp. 680–692, Apr. 2018.

[30] S. Baldi, S. Roy, and K. Yang
Towards adaptive autopilots for fixed-wing unmanned aerial vehicles
In *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 4724–4729.

[31] Z. S. Hou
The parameter identification, adaptive control and model free learning adaptive control for nonlinear systems
Ph.D. dissertation, Dept. Autom. Control, Northeastern Univ., Shenyang, China, 1994.

[32] Z. Hou and S. Xiong
On model-free adaptive control and its stability analysis
IEEE Trans. Autom. Control, vol. 64, no. 11, pp. 4555–4569, Nov. 2019.

[33] S. A. Hashjin, S. Pang, E.-H. Miliani, K. Ait-Abderrahim, and B. Nahid-Mobarakeh
Data-driven model-free adaptive current control of a wound rotor synchronous machine drive system
IEEE Trans. Transport. Electrific., vol. 6, no. 3, pp. 1146–1156, Sep. 2020.

[34] H. Gao, G. Ma, Y. Lv, and Y. Guo
Forecasting-based data-driven model-free adaptive sliding mode attitude control of combined spacecraft
Aerosp. Sci. Technol., vol. 86, no. MAR., pp. 364–374, 2019.

[35] Y. Liao, Q. Jiang, T. Du, and W. Jiang
Redefined output model-free adaptive control method and unmanned surface vehicle heading control
IEEE J. Ocean. Eng., vol. 45, no. 3, pp. 714–723, Jul. 2020.

[36] M. Khosravi and A. B. Novinzadeh
A multi-body control approach for flapping wing micro aerial vehicles
Aerosp. Sci. Technol., vol. 112, no. 2, 2021, Art. no. 106525.

[37] S. Liu, Z. Hou, T. Tian, Z. Deng, and Z. Li
A novel dual successive projection-based model-free adaptive control method and application to an autonomous car
IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 11, pp. 3444–3457, Nov. 2019.

[38] Y. Jiang, X. Xu, and L. Zhang
Heading tracking of 6WID/4WIS unmanned ground vehicles with variable wheelbase based on model free adaptive control
Mech. Syst. Signal Process., vol. 159, no. 4, 2021, Art. no. 107715.

[39] M. F. Heertjes, B. Van der Velden, and T. Oomen
Constrained iterative feedback tuning for robust control of a wafer stage system
IEEE Trans. Control Syst. Technol., vol. 24, no. 1, pp. 56–66, Jan. 2016.

[40] L. Duan, Z. Hou, X. Yu, S. Jin, and K. Lu
Data-driven model-free adaptive attitude control approach for launch vehicle with virtual reference feedback parameters tuning method
IEEE Access, vol. 7, pp. 54106–54116, 2019.

[41] S. Baldi, G. Battistelli, D. Mari, E. Mosca, and P. Tesi
Multi-model adaptive switching control with fine controller tuning
IFAC Proc. Volumes, vol. 44, no. 1, pp. 374–379, 2011.

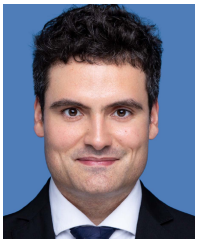
[42] J. Yang, X. Wang, S. Baldi, S. Singh, and S. Farì
A software-in-the-loop implementation of adaptive formation control for fixed-wing UAVs
IEEE/CAA J. Automatica Sinica, vol. 6, no. 5, pp. 1230–1239, Sep. 2019.

[43] S. Farì
Guidance and control for a fixed-wing UAV
M. S. thesis, Dept. Automat. Control Eng., Politecnico di Milano, Milan, Italy, 2017.

[44] Automatic tuning with autotune
[Online]. Available: https://ardupilot.org/plane/docs/automatic-tuning-with-autotune.html

[45] T. Espinoza, A. Dzul, R. Lozano, and P. Parada
Backstepping - sliding mode controllers applied to a fixed-wing UAV
In *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2013, pp. 95–104.

[46] A. E. Fraire, Y. Chen, A. Dzul, and R. Lozano
Fixed-wing mav adaptive PD control based on a modified mit rule with sliding-mode control
In *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2017, pp. 1647–1656.



Simone Baldi (Senior Member, IEEE) received the B.Sc. degree in electrical engineering in 2005, the M.Sc. degree in control engineering in 2005, and the Ph.D. degree in automatic control engineering in 2011, from the University of Florence, Florence, Italy.

He is currently a Professor with the School of Mathematics, Southeast University, Nanjing, China, and guest position with Delft Center for Systems and Control, TU Delft, the Netherlands. His research interests include adaptive and learning systems and intelligent vehicles.

Dr. Baldi was awarded outstanding reviewer for *Automatica* in 2017. He is a Subject Editor of the *International Journal of Adaptive Control and Signal Processing* and an Associate Editor of the *IEEE Control Systems Letters*.

Danping Sun received the B.Sc. degree in information science from the Hubei University of Science and Technology, Xianning, China, in 2015, and is currently working toward the postgraduate degree in electronic and electrical engineering with Wuhan Textile University, Wuhan, China.

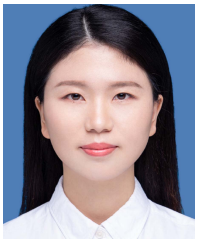
Her research interests include adaptive systems and unmanned vehicles.

Xin Xia (Graduate Student Member, IEEE) received the B.Sc. degree in electrical engineering and automation from the Wuhan Institute of Technology, Wuhan, China, in 2014, and the M.Sc. degree in control engineering from the Kunming University of Science and Technology, Kunming, China, in 2017. He is currently working toward the Ph.D. degree in mathematics with Southeast University, Nanjing, China.

His research interests include adaptive systems and unmanned vehicles.

Guopeng Zhou received the B.Sc. degree in mathematics from Hubei University, Wuhan, China, in 1994, and the M.Sc. and Ph.D. degrees in automation from the South China University of Technology, Guangzhou, China, in 2005 and 2008, respectively.

He is currently the Director of Hubei Electrical Machinery and Control System Engineering Technology Research Center, Xianning, China, and also the Xianning Intelligent Electromechanical Industry Alliance Responsible Person. His research interests include multiagent systems, adaptive systems, intelligent mechanical and electrical systems, and fault diagnosis.



Di Liu (Member, IEEE) received the B.Sc. degree in information science from the Hubei University of Science and Technology, Xianning, China, and the M.Sc. degree in control science and engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2014 and 2017, respectively, and the Ph.D. degree in cyber science and engineering from Southeast University, Nanjing, China, in 2021. She is currently working toward the second Ph.D. degree in systems and control with the Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Groningen, The Netherlands.

Her research interests include learning systems and control, with application in intelligent transportation, automated vehicles, and unmanned aerial vehicles.