Control Engineering Practice 67 (2017) 76–91

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Path-following control of small fixed-wing unmanned aircraft systems with $\mathcal{H}_\infty$ type performance

Devaprakash Muniraj, Mark C. Palframan, Kyle T. Guthrie, Mazen Farhood *

Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061, United States

ARTICLE INFO

Keywords:
Unmanned aircraft system
Fixed-wing aircraft
H-infinity control
Path following
Linear parameter-varying control

ABSTRACT

The focus of this paper is on the design of path-following controllers for a small fixed-wing unmanned aircraft system (UAS) using the $\mathcal{H}_\infty$ robust control framework. The robust controllers are synthesized based on a lumped path-following and UAS dynamics formulation, effectively combining the six degree-of-freedom aircraft dynamics with the established parallel transport frame virtual vehicle dynamics. Two path-following controllers with a conventional cascaded architecture consisting of an outer guidance loop and an inner stabilization loop are also considered as points of reference. The robustness and performance of these controllers are tested in a rigorous MATLAB simulation environment that includes steady winds, turbulence, measurement noise, and time delays. Finally, flight experiments are conducted on a small fixed-wing UAS platform, and the controllers are compared in terms of tracking performance, control effort, and ease of implementation.

1. Introduction

One of the challenges associated with small fixed-wing unmanned aircraft systems (UAS) is operating effectively in the presence of relatively significant environmental disturbances, namely winds, gusts, and turbulence. Owing to their size, small UAS are affected much more dramatically than traditional aircraft in the presence of what may be considered small atmospheric disturbances, often leading to poor performance by traditional trajectory tracking methods with time-stamped inertial position feedback. Path-following control methods, by guiding an aircraft to converge to and follow a geometric path in space specified without time parametrization, can potentially handle stronger disturbances when compared to trajectory tracking methods (Dačić, 2005). In fact, Aguiar, Hespanha, and Kokotović (2007) shows that path-following control is often able to remove the inherent performance limitations present in reference tracking methods.

Path-following control has been shown to have many useful applications for UAS, primarily involving missions related to surveillance, imaging, formation flight, and station keeping (Kaminer, Yakimenko, Dobrokhodov, Pascoal, Hovakimyan, Cao, Young, Patel, 2007; Rysdyk, 2006; Wise & Rysdyk, 2006). Flying various loiter patterns, for instance, is necessary to collect sparsely distributed airborne containments (Palframan & Woolsey, 2014). UAS operating in urban environments must maintain their position on the desired path despite the presence of significant disturbances in order to avoid collisions with the surrounding infrastructure.

Some notable approaches to path following include proportional navigation inspired nonlinear guidance logic (Park, Deyst, & How, 2004), waypoint guidance (Osborne & Rysdyk, 2005), and the use of vector fields to generate course commands to drive a vehicle towards the desired path (Nelson, Barber, McLain, & Beard, 2006; Sujit, Saripalli, & Sousa, 2014). Among others, this work utilizes a virtual vehicle formulation (Kaminer, Pascoal, Xargay, Hovakimyan, Cao, & Dobrokhodov, 2010; Soetanto, Lapierre, & Pascoal, 2003), whereby the controller strives to minimize the error between the ownship and a fictitious vehicle constrained to move along the path. Specifically, this work strives to extend the virtual vehicle formulation in Kaminer et al. (2010) by incorporating it with the UAS dynamics, resulting in a combined path-following and UAS dynamics formulation. At the cost of increased complexity through the introduction of path-following dynamics into the overall system, a virtual vehicle approach can be considered the most flexible path-following approach in terms of the geometric paths that can be followed. In fact, the allowable paths include any curvature-bounded geometrically defined path. By incorporating key assumptions into the approach developed in Kaminer et al. (2010), the approach in this work effectively combines the path-following dynamics and aircraft dynamics in a way that the resulting system has the same number of

* Corresponding author.
E-mail addresses: devapm@vt.edu (D. Muniraj), palframan@vt.edu (M.C. Palframan), kguthrie@vt.edu (K.T. Guthrie), farhood@vt.edu (M. Farhood).

http://dx.doi.org/10.1016/j.conengprac.2017.07.006
Received 24 February 2017; Received in revised form 14 July 2017; Accepted 14 July 2017
Available online 12 August 2017
0967-0661/

states as the original aircraft dynamics, thereby minimizing the cost of using the virtual vehicle framework.

While much of the existing literature on path-following control methods employs either nonlinear control, such as backstepping, sliding mode, and adaptive control, e.g., see Kaminer et al. (2010); Yang and Kim (1999), or proportional–integral–derivative (PID) control, a robust control approach is used in this work. The performance guarantees against plant uncertainties and exogenous disturbances provided by robust controllers make them an appropriate choice for application to path-following control of small UAS. The combined nonlinear path-following vehicle dynamics are linearized about an aircraft trim and parameterized by the path curvature to result in a linear parameter-varying (LPV) system. Then, two controllers of varying complexity, namely a linear time-invariant (LTI) controller and an LPV controller, are designed using the ℓ_2-induced norm as the performance measure. This linear framework enables us to take advantage of tools from the vast literature on robust control. For instance, the formulation presented within is easily adaptable to formal validation techniques and the incorporation of uncertain initial conditions and other uncertainties into the synthesis process (Arifianto & Farhood, 2015b; Palframan & Farhood, 2016). In general, the control strategy developed in this paper utilizes the fact that control of the vehicle attitude can drive the vehicle towards a desired position, in this case along a geometric path in space. The controllers need only a curvature-bounded geometrically defined path as input to accurately track the path at a constant airspeed in the midst of atmospheric and other disturbances. As the controllers are not path dependent, they can be synthesized offline.

In order to compare the proposed method with the existing cascaded approach for path following, two path-following controllers consisting of an outer guidance loop and an inner stabilization loop are also presented herein. The first controller is partially based on the virtual vehicle formulation proposed by Kaminer et al. (2010), where a nonlinear backstepping outer-loop controller, based on the parallel transport frame dynamics, prescribes pitch and yaw rate commands, which are then tracked by an $\mathcal{L}_1$ adaptive controller. In this work, however, a standard $\mathcal{H}_\infty$ controller is used instead of the $\mathcal{L}_1$ adaptive controller as the stabilizing inner-loop controller. The other cascaded controller is based on the path-following controller available in the open-source ArduPilot software (ArduPilot, 2016a). It utilizes a nonlinear guidance logic based on Park et al. (2004) and has four PID controllers in the inner loop for stabilization.

All controllers are designed based on a nonlinear model of a Senior Telemaster UAS (Hobby Express, 2017) developed from flight test data. Extensive simulation in a rigorous MATLAB environment is performed to evaluate the path-following performance of each controller. The controllers are then implemented on the Senior Telemaster UAS and flight experiments are conducted under different environmental conditions to validate the findings from the simulation studies.

This paper builds upon the work in Palframan, Guthrie, and Farhood (2015), which in turn is based on the work found in Guthrie (2013). The work presented in this paper differs from the conference paper Palframan et al. (2015) in the following respects: (i) the airframe considered in this study is different from the 6-ft Telemaster aircraft considered in Palframan et al. (2015), and consequently the aerodynamic and propulsion models as well as the controllers presented in this work are different; (ii) this paper provides a more detailed explanation of the control synthesis procedures; (iii) an additional path-following controller similar to the one used in the commercially available and widely used autopilot software, ArduPilot, is considered herein; (iv) in this paper, in addition to simulation studies, the performances of the controllers are compared through extensive flight tests conducted on a small fixed-wing UAS platform; and (v) the technical challenges associated with the real-time implementation of the controllers on the UAS platform are also addressed. The paper is organized as follows. Sections 2 and 3 cover the background information for fixed-wing UAS and parallel transport frame path-following dynamics, respectively. Linear plant model formulation and the controller synthesis procedures for the four controllers are presented in Section 4. Next, the simulation environment and performance metrics used are presented in Section 5. Section 6 describes the UAS experimental platform including the onboard sensors, airframe, and the system architecture. Simulation and flight test results for various geometric paths are presented, and the relative merits and limitations of each controller are discussed in Section 7. Finally, conclusions and possible topics of future work are provided in Section 8.

The notation used is quite standard. The set of real numbers is denoted by $\mathbb{R}$ and that of real $n \times 1$ vectors by $\mathbb{R}^n$. $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix and $\boldsymbol{0}_{n\times m}$ denotes an $n \times m$ matrix with zero entries. If $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N$ are matrices, then $\mathrm{diag}(\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N)$ denotes their block-diagonal augmentation. Given a symmetric matrix $\boldsymbol{X}$, $\boldsymbol{X} \prec \boldsymbol{0}$ indicates that X is negative definite. The normed space of square summable vector-valued sequences is denoted by ℓ_2. It consists of elements $\boldsymbol{w} = (\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots)$, with each $\boldsymbol{w}_i \in \mathbb{R}^n$ for some n, having a finite norm $\|\boldsymbol{w}\|_{\ell_2}$ defined by $\|\boldsymbol{w}\|_{\ell_2}^2 = \sum_{i=0}^{\infty} \boldsymbol{w}_i{}^T \boldsymbol{w}_i$. Finally, the notation $\|\boldsymbol{G}\|_{\ell_2\to\ell_2}$ denotes the ℓ_2-induced norm of a bounded linear mapping $\boldsymbol{G}$ on ℓ_2.

2. UAS flight dynamic model

The UAS dynamic model and controllers in this work are based on a commercially available Senior Telemaster radio-controlled aircraft from Hobby Express operated with a customized Pixhawk autopilot system.

2.1. Rigid-body equations of motion

Detailed derivations of the rigid-body equations of motion for a fixed-wing aircraft are available in Raol and Singh (2009), among others. Two reference frames are used to define the aircraft's motion, the Earth-fixed inertial reference frame and the body-fixed reference frame, denoted by $\mathcal{F}_I$ and $\mathcal{F}_b$, respectively. The inertial reference frame has its origin on the surface of the Earth and has components of $(\boldsymbol{x}_I, \boldsymbol{y}_I, \boldsymbol{z}_I)$, which point North, East, and downwards, respectively. The position vector from the origin of $\mathcal{F}_I$ to the nominal aircraft center of gravity (CG) is defined as $\boldsymbol{p} = [N, E, z_g]^T$, where N is the north component, E is the east component, and $-z_g$ is the altitude above ground level (AGL). The body frame $\mathcal{F}_b$, as shown in Fig. 1, has its origin affixed to the aircraft CG, and has components of $(\boldsymbol{x}_b, \boldsymbol{y}_b, \boldsymbol{z}_b)$, which point towards the aircraft nose, towards the right wingtip, and downwards, respectively. The wind reference frame $\mathcal{F}_w$ has its origin at the aircraft CG and is obtained by a left-hand rotation about $\boldsymbol{y}_b$ by the angle of attack α, followed by a right-hand rotation about the resulting $\boldsymbol{z}$-axis by the sideslip angle β, as shown in Fig. 1.

The gravitational acceleration vector $\boldsymbol{g}$ expressed in $\mathcal{F}_b$ has components $g[-\sin\theta, \cos\theta\sin\phi, \cos\theta\cos\phi]^T$ with $g = 9.80665\,\mathrm{m/s}^2$. The roll, pitch, and yaw Euler angles are denoted as $\lambda = [\phi, \theta, \psi]^T$. The linear velocity of $\mathcal{F}_b$ with respect to $\mathcal{F}_I$ expressed in the body frame is denoted as $\boldsymbol{v} = [u, v, w]^T$. Likewise, the angular velocities about $(\boldsymbol{x}_b, \boldsymbol{y}_b, \boldsymbol{z}_b)$ are denoted as $\boldsymbol{\omega} = [p, q, r]^T$. The state vector of the aircraft can then be defined as $\boldsymbol{x} = [\boldsymbol{\omega}^T, \boldsymbol{v}^T, \lambda^T, \boldsymbol{p}^T]^T$. The linear velocity of the aircraft relative to the wind is given by $\bar{\boldsymbol{v}} = \boldsymbol{v} - \boldsymbol{v}_w$, where $\boldsymbol{v}_w = [u_w, v_w, w_w]^T$ is the wind velocity relative to $\mathcal{F}_I$ expressed in $\mathcal{F}_b$ and denotes the exogenous disturbances due to atmospheric turbulence and steady wind. The airspeed, denoted by V_a, angle of attack α, and angle of sideslip β are defined as

$$V_a = (\bar{\boldsymbol{v}}^T \bar{\boldsymbol{v}})^{\frac{1}{2}}, \quad \alpha = \tan^{-1}\frac{w - w_w}{u - u_w}, \quad \text{and} \quad \beta = \sin^{-1}\frac{v - v_w}{V_a}.$$

Given the symmetry resulting from the choice of the body-fixed reference frame, the inertia tensor can be simplified by taking $I_{xy} = I_{yx} = I_{yz} = I_{zy} = 0$. Additionally, I_{xz} and I_{zx} are assumed to be negligibly small. The remaining inertia terms are determined by

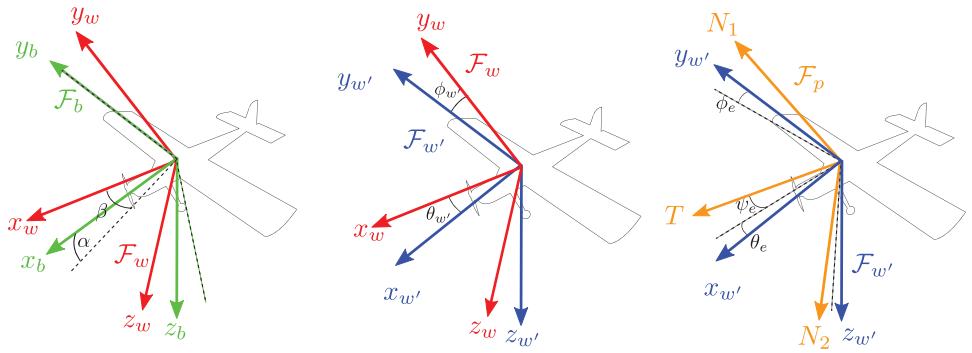


Fig. 1. The body-fixed reference frame, $\mathcal{F}_b$, wind reference frame, $\mathcal{F}_w$, and parallel transport frame, $\mathcal{F}_p$, for an aircraft.

compound pendulum tests resulting in the inertia matrix (with units kg-m^2)

$$\boldsymbol{J}_b = \begin{bmatrix} 1.32 & 0 & 0 \\ 0 & 1.57 & 0 \\ 0 & 0 & 1.87 \end{bmatrix}.$$

Let $\boldsymbol{\delta} = [\delta_E, \delta_A, \delta_R, \delta_T]^T$ denote the input vector of the aircraft, where δ_E, δ_A, and δ_R are the elevator, aileron, and rudder deflections, respectively, and δ_T is the throttle input. The net external force and moment acting on the aircraft expressed in $\mathcal{F}_b$ are denoted as $\boldsymbol{f}(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta})$ and $\boldsymbol{m}(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta})$, respectively. The map $\mathcal{F}_b \mapsto \mathcal{F}_I$ is defined by the rotation matrix $\boldsymbol{\mathcal{R}}_{Ib}$, which is given as

$$\boldsymbol{\mathcal{R}}_{Ib}(\lambda) = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}.$$

The differential equations representing the aircraft motion can then be expressed as

$$\dot{\boldsymbol{v}} = m^{-1}\boldsymbol{f}(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) + \boldsymbol{g} - \boldsymbol{\omega} \times \boldsymbol{v}, \tag{1a}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}_b^{-1}\boldsymbol{m}(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) - \boldsymbol{J}_b^{-1}(\boldsymbol{\omega} \times \boldsymbol{J}_b\boldsymbol{\omega}), \tag{1b}$$

$$\dot{\lambda} = \boldsymbol{E}(\phi, \theta)\boldsymbol{\omega}, \tag{1c}$$

$$\dot{\boldsymbol{p}} = \boldsymbol{\mathcal{R}}_{Ib}(\lambda)\boldsymbol{v}, \tag{1d}$$

where

$$\boldsymbol{E}(\phi, \theta) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{2}$$

and $m = 5.71$ kg is the aircraft mass. Note that a singularity exists in the equations of motion when the aircraft is at $\theta = \pm 90°$. Since the aircraft does not experience such steep attitudes in this work, the singularity does not pose a problem.

It is found that changes in the initial heading direction (ψ_0) has a large effect on the linearization of $\dot{\boldsymbol{p}}$. To circumvent this, the nominal CG position is redefined as $\boldsymbol{p} = [X, Y, z_g]^T$, where $X = N\cos\psi_0 + E\sin\psi_0$ and $Y = -N\sin\psi_0 + E\cos\psi_0$. Eq. (1d) is then rewritten as

$$\dot{\boldsymbol{p}} = \boldsymbol{\mathcal{R}}_{Ib}(\lambda_0)\boldsymbol{v}, \tag{3}$$

where $\lambda_0 = [\phi, \theta, \psi - \psi_0]^T$.

2.2. Aerodynamic and propulsion models

The aerodynamic and propulsive forces and moments are modeled separately, and as such, are written as

$$\boldsymbol{f}(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) = \boldsymbol{f}_A(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) + \boldsymbol{f}_P(\bar{\boldsymbol{v}}, \delta_T),$$
$$\boldsymbol{m}(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) = \boldsymbol{m}_A(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) + \boldsymbol{m}_P(\bar{\boldsymbol{v}}, \delta_T),$$

where $(\cdot)_A$ and $(\cdot)_P$ denote the aerodynamic and propulsive components, respectively. In this work, the thrust force is generated by an electric motor driven propeller and is assumed to act along the $\boldsymbol{x}_b$ axis. Furthermore, propeller effects including reaction torque, P-factor, propwash, and gyroscopic precession are not modeled. These assumptions allow us to define $\boldsymbol{m}_P(\bar{\boldsymbol{v}}, \delta_T) = 0$ and $\boldsymbol{f}_P(\bar{\boldsymbol{v}}, \delta_T) = [T(V_a, \delta_T), 0, 0]^T$, where $T(\cdot)$ is the generated propeller thrust. It is convenient to define $\boldsymbol{f}_A(\cdot) = [f_x(\cdot), f_y(\cdot), f_z(\cdot)]^T$ and $\boldsymbol{m}_A(\cdot) = [m_l(\cdot), m_m(\cdot), m_n(\cdot)]^T$.

Assuming that the electronic speed controller provides a constant propeller RPM for a given input command δ_T, an experimentally obtained lookup table is used to map δ_T to a corresponding RPM value following a procedure similar to the one explained in Arifianto and Farhood (2015a). A blade element momentum theory based propeller analysis tool Javaprop (Hepperle, 2006) is then used to obtain a lookup table which maps propeller RPM and airspeed to propeller thrust. Finally, the dynamics of the propulsion system are modeled as a second-order system with the following values for the natural frequency and damping ratio:

$$\omega_n = \begin{cases} 5.7\,\text{rad/s} & \text{for } \dot{\delta}_T \geq 0 \\ 4.5\,\text{rad/s} & \text{for } \dot{\delta}_T < 0, \end{cases} \qquad \zeta = \begin{cases} 1.3 & \text{for } \dot{\delta}_T \geq 0 \\ 1.6 & \text{for } \dot{\delta}_T < 0, \end{cases}$$

where the above values are determined experimentally for zero airspeed.

The aerodynamic forces and moments are defined in terms of dimensionless aerodynamic coefficients as

$$f_i(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) = \frac{1}{2}C_i(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta})\rho V_a^2 Q, \text{ for } i = x, y, z,$$
$$m_j(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) = \frac{1}{2}C_j(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta})\rho V_a^2 Q b, \text{ for } j = l, n,$$
$$m_m(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta}) = \frac{1}{2}C_m(\bar{\boldsymbol{v}}, \boldsymbol{\omega}, \boldsymbol{\delta})\rho V_a^2 Q \bar{c},$$

where $C_{(\cdot)}$ denotes an aerodynamic coefficient, ρ is the air density, $Q = 0.86\,\text{m}^2$ is the wing area, $b = 2.39$ m is the wingspan, and $\bar{c} = 0.36$ m is the mean aerodynamic chord. The aerodynamic coefficients are estimated using the output error method with the following model structure:

$$C_x = C_{x_0} + C_{x_\alpha}\alpha + C_{x_{\delta_T}}\delta_T + C_{x_T}\frac{2T}{\rho S V_a^2},$$
$$C_y = C_{y_0} + C_{y_\beta}\beta + C_{y_{\delta_A}}\delta_A + C_{y_{\delta_R}}\delta_R + (C_{y_p}p + C_{y_r}r)\frac{b}{2V_a},$$
$$C_z = C_{z_0} + C_{z_\alpha}\alpha + C_{z_{\delta_E}}\delta_E + C_{z_q}\frac{q\bar{c}}{2V_a} + C_{z_T}\frac{2T}{\rho S V_a^2},$$
$$C_l = C_{l_0} + C_{l_\beta}\beta + C_{l_{\delta_A}}\delta_A + C_{l_{\delta_R}}\delta_R + (C_{l_p}p + C_{l_r}r)\frac{b}{2V_a},$$
$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_E}}\delta_E + C_{m_q}\frac{q\bar{c}}{2V_a},$$
$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_{\delta_A}}\delta_A + C_{n_{\delta_R}}\delta_R + (C_{n_p}p + C_{n_r}r)\frac{b}{2V_a}. \tag{4}$$

The aerodynamic parameter values are estimated from flight tests, where automated inputs are applied to each of the control surfaces.

Table 1
Aerodynamic Parameter Values.

Term	Value	Term	Value	Term	Value	Term	Value	Term	Value	Term	Value
C_{x_0}	-0.1303	C_{y_0}	0.0165	C_{z_0}	-0.5892	C_{l_0}	0.0019	C_{m_0}	-0.1078	C_{n_0}	-0.0015
C_{x_α}	-0.0224	C_{y_β}	-0.3816	C_{z_α}	-3.8864	C_{l_β}	-0.0582	C_{m_α}	-0.6969	C_{n_β}	0.0451
C_{x_T}	-0.0195	$C_{y_{\delta_A}}$	-0.0413	$C_{z_{\delta_E}}$	0.3061	$C_{l_{\delta_A}}$	0.1619	$C_{m_{\delta_E}}$	0.9405	$C_{n_{\delta_A}}$	-0.0169
$C_{x_{\delta_T}}$	0.1028	$C_{y_{\delta_R}}$	-0.1196	C_{z_T}	1.4463	$C_{l_{\delta_R}}$	-0.0128	C_{m_q}	-11.8094	$C_{n_{\delta_R}}$	0.0555
		C_{y_p}	0.1445	C_{z_q}	-1.9859	C_{l_p}	-0.4137			C_{n_p}	-0.0291
		C_{y_r}	0.1441			C_{l_r}	0.1815			C_{n_r}	-0.0781

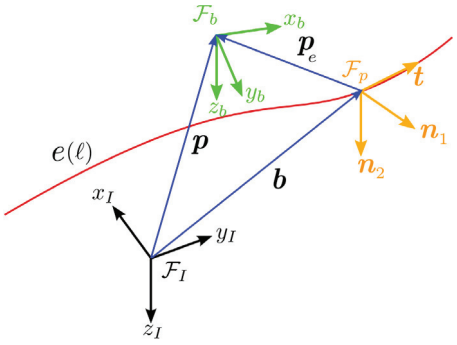


Fig. 2. The parallel transport frame is related to the current UAS position by the error vector p_e.

The flight tests are performed in the presence of minimum atmospheric disturbances as the output error method does not take into account process noise. For more details on aerodynamic modeling, the reader is referred to Arifianto and Farhood (2015a), where a similar procedure is adopted. The aerodynamic parameter values for the UAS platform are provided in Table 1.

2.3. Servo models

The three servomotors used to deflect the control surfaces are modeled as three identical second-order systems. The natural frequency and damping ratio of the second-order model are experimentally obtained from frequency response tests, and the values, $\omega_n = 13.7\,\text{rad/s}$ and $\zeta = 0.67$, are found to result in a good fit to the collected frequency response data. The interested reader is referred to Arifianto and Farhood (2015a) for more details on the experiment. The servo models map the control surface commands $(\delta_{E_c}, \delta_{A_c}, \delta_{R_c})$ to the corresponding control surface deflections $(\delta_E, \delta_A, \delta_R)$.

3. Formulation of the path-following problem

Much of the background, definitions, and formulation of the path-following problem are borrowed from Kaminer et al. (2010), with some modifications made for simplification or preference. Let $e(\ell)$ represent the path to be followed in $\mathcal{F}_I$, parameterized by the path length ℓ. The path-following dynamics are based on a virtual vehicle moving along the given path at some prescribed rate. At each point on the path, the virtual vehicle has an associated reference frame called the parallel transport frame (sometimes referred to as a rotation minimizing frame) (Bishop, 1975; Hanson & Ma, 1995), denoted $\mathcal{F}_p$, affixed to the virtual vehicle CG. The three orthonormal basis vectors of $\mathcal{F}_p$, denoted $t(\ell)$ (tangent vector), $n_1(\ell)$ (first normal vector), and $n_2(\ell)$ (second normal vector), satisfy the following equations:

$$\begin{bmatrix} dt(\ell)/d\ell \\ dn_1(\ell)/d\ell \\ dn_2(\ell)/d\ell \end{bmatrix} = \begin{bmatrix} 0 & k_1(\ell) & k_2(\ell) \\ -k_1(\ell) & 0 & 0 \\ -k_2(\ell) & 0 & 0 \end{bmatrix} \begin{bmatrix} t(\ell) \\ n_1(\ell) \\ n_2(\ell) \end{bmatrix}, \tag{5}$$

where $k_1(\ell)$ and $k_2(\ell)$ are parameters that vary over ℓ. Let $p_e = [x_e, y_e, z_e]^T$ be the position error vector between the UAS and the virtual vehicle, expressed in $\mathcal{F}_p$, as shown in Fig. 2. Also, define $\mathcal{F}_{w'}$ as the rotation of the UAS wind reference frame $\mathcal{F}_w$ onto the local level plane, as shown in Fig. 1. $\mathcal{F}_{w'}$ can be described relative to $\mathcal{F}_p$ through a set of Euler angle errors given by $\lambda_e = [\phi_e, \theta_e, \psi_e]^T$. Using a small angle approximation, it is assumed that the angular rates in $\mathcal{F}_{w'}$ are approximately equal to ω. After differentiation and simplification, the following equations are obtained for the kinematic position error dynamics, $\dot{p}_e$, of the combined UAS and virtual vehicle system:

$$\begin{aligned} \dot{x}_e &= -\dot{\ell}(1 - k_1(\ell)y_e - k_2(\ell)z_e) + V_a \cos\theta_e \cos\psi_e, \\ \dot{y}_e &= -\dot{\ell}k_1(\ell)x_e + V_a \cos\theta_e \sin\psi_e, \\ \dot{z}_e &= -\dot{\ell}k_2(\ell)x_e - V_a \sin\theta_e. \end{aligned} \tag{6}$$

The attitude error dynamics, $\dot{\lambda}_e$, can be derived in a similar fashion resulting in the following equations:

$$\begin{aligned} \dot{\phi}_e &= p + (\dot{\ell}k_2(\ell)\sin\psi_e + r\cos\phi_e \sin\theta_e + q\sin\phi_e \sin\theta_e)/\cos\theta_e, \\ \dot{\theta}_e &= \dot{\ell}k_2(\ell)\cos\psi_e + q\cos\phi_e - r\sin\phi_e, \\ \dot{\psi}_e &= -\dot{\ell}k_1(\ell) + (\dot{\ell}k_2(\ell)\sin\theta_e \sin\psi_e + q\sin\phi_e + r\cos\phi_e)/\cos\theta_e. \end{aligned} \tag{7}$$

Together, (6) and (7) describe the path-following error dynamics of the combined UAS and virtual vehicle system. Finally, the dynamics of the virtual vehicle are defined as

$$\dot{\ell} = K_1 x_e + V_a \cos\theta_e \cos\psi_e, \tag{8}$$

where K_1 is a positive constant, chosen to be 2 in this work.

Following the example set in Kaminer et al. (2010), the error Euler angles, θ_e and ψ_e, are shaped using approach angle functions to improve control performance. In a departure from the method used in Kaminer et al. (2010), hyperbolic tangent functions are used as the approach angle functions for convenience. The approach angles work to ensure that the vehicle is approaching the path at all times, and provide an extra degree of freedom in the aggressiveness of the tracking behavior. The approach angles, θ_δ and ψ_δ, are defined as

$$\theta_\delta(z_e) = \theta_m \tanh\left(z_e/C_1\right), \quad \psi_\delta(y_e) = \psi_m \tanh\left(-y_e/C_2\right),$$

where θ_m and ψ_m are the maximum desired approach angles, and C_1 and C_2 are scaling factors used to determine the magnitude of the position error corresponding to the maximum allowed approach angle. To incorporate these shaping functions, λ_e is redefined as $[\phi_e, \theta_e - \theta_\delta, \psi_e - \psi_\delta]^T$. A sample shaping function for altitude error is shown in Fig. 3 with $C_1 = 8$ and $\theta_m = 20°$. Note that the approach angle is zero when there is no altitude error, and saturates at $\pm\theta_m$ when the altitude error is large.

For this work, the set of permissible geometric paths to be followed is restricted to 2-D paths in the x_I-y_I plane. For this special case, many simplifications can be made to the relevant system dynamics. The largest simplification is that the UAS pitch and bank angles can be considered identically equal to the previously defined error angles θ_e and ϕ_e. Additionally, the $k_2(\ell)$ path parameter will be identically zero for all time, allowing $\dot{p}_e$ and $\dot{\lambda}_e$ to be simplified. Given $k_2(\ell) = 0$, the remaining parallel transport frame parameter $k_1(\ell)$ can be ascribed a more physical interpretation, namely, the inverse of the current radius of curvature of the path $R(\ell)$ (Hanson & Ma, 1995), that is,

$$k_1(\ell) = \frac{1}{R(\ell)}. \tag{9}$$

A straight line path therefore corresponds to an infinite radius of curvature and a parameter value of $k_1 = 0$. As the radius of curvature gets smaller and the corresponding turn gets tighter, the magnitude of k_1 increases.

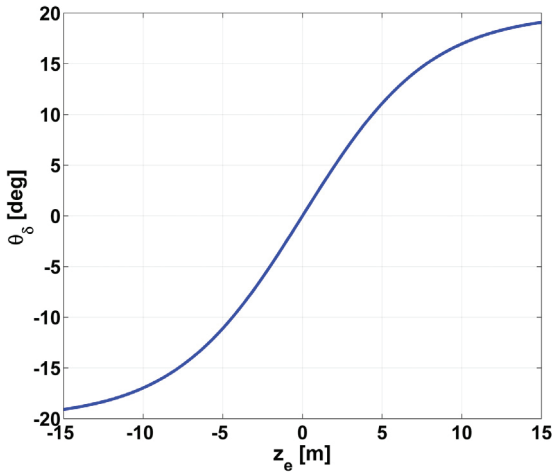


Fig. 3. A sample approach angle shaping function guides the aircraft to the correct altitude and saturates at $\theta_\delta = 20°$.

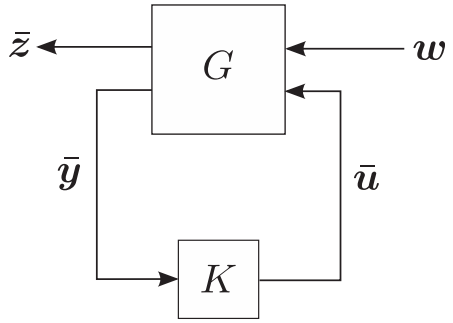


Fig. 4. The closed-loop LTI system.

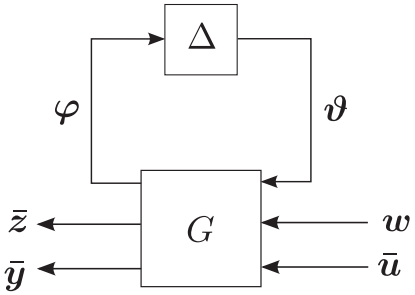


Fig. 5. The open-loop LFR system.

4. Controller synthesis

In this section, four different types of controllers for path following are presented. Three of the controllers are model-based controllers synthesized based on a linear model of the system G, shown in Figs. 4 and 5. Three linearized models are developed for the synthesis of the three model-based controllers. An LPV model with a polynomial dependence on k_1 is first developed from the UAS and path-following dynamic equations, and an LPV controller is synthesized. The dependence of the parameter k_1 on the path location ℓ is dropped for simplicity of notation. An LTI plant model is then formulated by setting $k_1 = 0$ in the LPV model. Finally, a second LTI model is developed based on the standard UAS equations of motion in order to synthesize an inner-loop $\mathcal{H}_\infty$ controller to track the pitch and yaw rates provided by an outer-loop backstepping controller. The fourth path-following controller is composed of a PID controller in the inner loop and a nonlinear guidance logic in the outer loop, and is based on the controller used in the commercially available ArduPilot (ArduPilot, 2016a).

4.1. LPV plant model

The lumped path-following and UAS system is composed of Eqs. (1a), (1b), (6) and (7). The state vector of the lumped system is defined as $\boldsymbol{x} = [\boldsymbol{v}^T, \boldsymbol{\omega}^T, \lambda_e^T, \boldsymbol{p}_e^T, \boldsymbol{x}_a^T]^T$, the control input as $\boldsymbol{u} = [\delta_{E_c}, \delta_{A_c}, \delta_{R_c}, \delta_T]^T$, the measurements as $\boldsymbol{y} = [\boldsymbol{\omega}^T, \lambda_e^T, V_a, \boldsymbol{p}_e^T]^T$, and the exogenous disturbances as $\boldsymbol{w} = [\boldsymbol{v}_w^T, \boldsymbol{w}_m^T]^T$, where $\boldsymbol{w}$ represents finite energy disturbances in $\mathcal{L}_2$. $\boldsymbol{w}_m$ represents measurement noise and is defined as $\boldsymbol{w}_m = [m_p, m_q, m_r, m_\phi, m_\theta, m_\psi, m_{V_a}, m_x, m_y, m_z]^T$, and $\boldsymbol{x}_a$ consists of the state variables of the actuators. The vectors $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$, $\boldsymbol{y}(t)$, and $\boldsymbol{w}(t)$ are real with dimensions denoted by n, n_u, n_y, and n_w, respectively. The equations of motion, performance, and measurement output can be written as $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}, k_1)$, $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u})$, and $\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{w})$, respectively.

Parameter-varying trim point

The equations of motion described above depend on k_1 through the path-following dynamic Eqs. (6) and (7). It is conceivable that parameterizing the trim states with respect to k_1, thereby making the trim states dependent on the path curvature, would result in a better approximation of the UAS nonlinear equations of motion as compared to linearizing the equations about straight and level flight. Assuming constant altitude flight, the trim bank angle $\phi_{e_{tr}}$ required to maintain a steady level turn of radius R and a tangential velocity V_t can be determined by relating the aircraft's lift and centripetal acceleration as

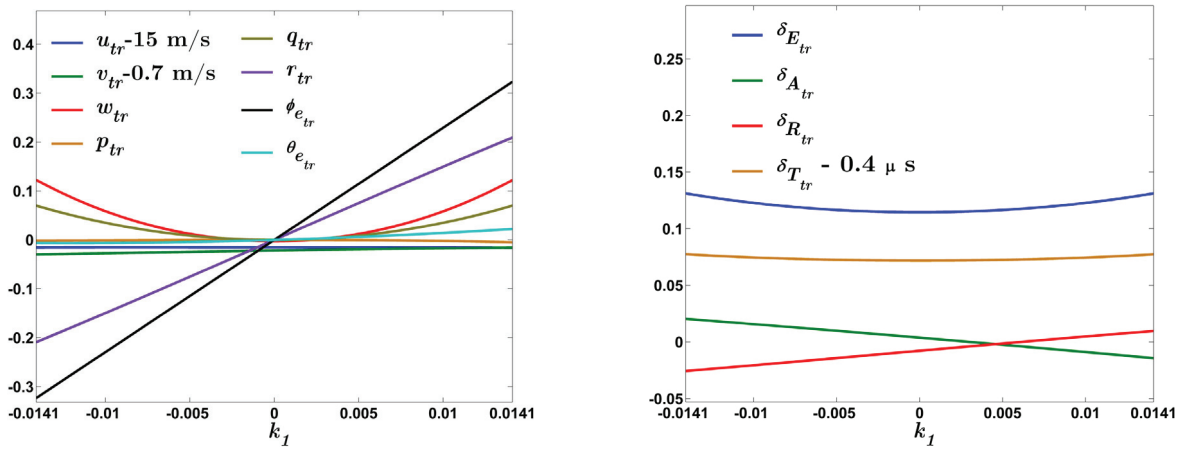
$$\tan \phi_{e_{tr}} = \frac{V_t^2}{gR}.$$

It is noted that $\boldsymbol{v}_w = \boldsymbol{0}$ for the chosen trim point, therefore, V_t is equivalent to $V_{a_{tr}}$, the desired airspeed of the UAS. Given the choice of bounded path curvatures, Eq. (9) and the small angle assumption are employed to solve for the trim bank angle as a function of k_1, namely

$$\phi_{e_{tr}}(k_1) = \frac{k_1 V_{a_{tr}}^2}{g}. \tag{10}$$

For a given k_1, the trim states and control inputs are determined by solving the following set of equations: $\dot{\boldsymbol{v}} = 0$, $\dot{\boldsymbol{\omega}} = 0$, $\dot{\phi}_e = 0$, $\dot{\theta}_e = 0$, $\dot{z}_e = 0$, $\phi_e - \phi_{e_{tr}} = 0$ and $V_a - V_{a_{tr}} = 0$, where $V_{a_{tr}} = 15$ m/s is the desired airspeed and $\phi_{e_{tr}}$ is determined using (10). The equations are solved using the `MATLAB` function `fsolve` over the range $-0.0141 \leq k_1 \leq 0.0141$. The computed trim states, measurements, and control inputs are given in Table 2 for straight and level flight ($k_1 = 0$), a moderate turn ($k_1 = \pm 0.009$), and an aggressive turn ($k_1 = \pm 0.0141$). The moderate turn with $k_1 = 0.009$ corresponds to a bank angle of approximately 12.0° for the chosen airspeed and a turn radius of 110.5 meters. The aggressive turn with $k_1 = 0.0141$ corresponds to a bank angle of approximately 18.5° and a turn radius of 70.9 meters. A more aggressive turn radius could not be used with the Senior Telemaster platform due to a limitation on the maximum achievable thrust, and hence k_1 is restricted to the interval $[-0.0141, 0.0141]$.

In order to express the LPV model as a linear fractional representation (LFR), the state–space equations should generally have rational dependence on the parameter k_1. The type of parameter dependence, be it linear, polynomial or rational, can affect the size of the resulting LFR and hence the computational complexity of the problem. With this in mind, the variation of the trim states and control inputs with respect to k_1, shown in Figs. 6a and 6b, are approximated by linear and quadratic functions over the range $-0.0141 \leq k_1 \leq 0.0141$. The associated linear



(a) Trim states. (b) Trim control inputs.

Fig. 6. Parameter-varying trim states and control inputs for the UAS platform. The units of $\{u_{tr}, v_{tr}, w_{tr}\}$, $\{p_{tr}, q_{tr}, r_{tr}\}$, $\{\phi_{e_{tr}}, \theta_{e_{tr}}\}$, and the PWM control input signals $\{\delta_{E_{tr}}, \delta_{A_{tr}}, \delta_{R_{tr}}, \delta_{T_{tr}}\}$ are meters per second, radians per second, radians, and microseconds, respectively.

Table 2
LPV Trim Points.

k_1	−0.0141	−0.009	0	0.009	0.0141
p	−0.001	−0.001	0.000	−0.002	−0.005
q	0.070	0.028	0.000	0.028	0.070
r	−0.209	−0.135	−0.001	0.135	0.209
u	14.984	14.985	14.984	14.984	14.984
v	0.670	0.673	0.678	0.683	0.684
w	0.123	0.047	−0.003	0.046	0.122
V_a	15.000	15.000	15.000	15.000	15.000
ϕ_e	−0.324	−0.207	0.000	0.207	0.324
θ_e	−0.007	−0.006	0.000	0.012	0.022
δ_E	0.131	0.121	0.115	0.121	0.131
δ_A	0.020	0.015	0.004	−0.008	−0.014
δ_R	−0.026	−0.019	−0.008	0.004	0.010
δ_T	0.478	0.474	0.472	0.474	0.477

Note: The units used for the angles, position, angular rates, velocities, and the PWM control input signals are radians, meters, radians per second, meters per second, and microseconds, respectively.

and quadratic functions for the trim states are given by

$$\begin{aligned} q_{tr}(k_1) &= 351.6886k_1^2 + 0.0007k_1 - 0.0001, \\ r_{tr}(k_1) &= 14.910k_1 - 0.0007, \\ \phi_{e_{tr}}(k_1) &= 22.9350k_1, \\ \theta_{e_{tr}}(k_1) &= 40.6673k_1^2 + 1.0254k_1 - 0.0002, \\ w_{tr}(k_1) &= 625.5609k_1^2 - 0.0123k_1 - 0.0035. \end{aligned} \tag{11}$$

The trim states $q_{tr}(k_1)$, $\theta_{e_{tr}}(k_1)$, and $w_{tr}(k_1)$ are approximated by quadratic functions, while $r_{tr}(k_1)$ and $\phi_{e_{tr}}(k_1)$ are approximated as linear functions of k_1. As expected, the bank angle $\phi_{e_{tr}}(k_1)$ and yaw rate $r_{tr}(k_1)$ have the largest variation with k_1. The trim states $u_{tr}(k_1)$, $v_{tr}(k_1)$, and $p_{tr}(k_1)$ are not strongly dependent on k_1 and are approximated by the values corresponding to straight and level flight. The errors due to the polynomial approximations are very small, with the RMS errors for the linear velocities, angular velocities, and attitude angles being less than 0.02 m/s, 1.6×10^{-3} rad/s, and 4.2×10^{-5} rad, respectively. The polynomial approximations for the control inputs are given by

$$\begin{aligned} \delta_{E_{tr}}(k_1) &= 83.8585k_1^2 - 0.0001k_1 + 0.1145, \\ \delta_{A_{tr}}(k_1) &= -1.2315k_1 + 0.0035, \\ \delta_{R_{tr}}(k_1) &= 1.2662k_1 - 0.0079. \end{aligned} \tag{12}$$

The lateral-directional control inputs $\delta_{A_{tr}}(k_1)$ and $\delta_{R_{tr}}(k_1)$ are approximated by linear functions, while the elevator control input $\delta_{E_{tr}}(k_1)$ is approximated by a quadratic function. The throttle control input $\delta_{T_{tr}}(k_1)$ is not strongly dependent on k_1 and is thus fixed at the trim value. The RMS values of the approximation errors for the control inputs are all less than 0.0027 μs.

Linearization and discretization

Linearizing the functions $\boldsymbol{f}(\cdot)$, $\boldsymbol{g}(\cdot)$, and $\boldsymbol{h}(\cdot)$ about the parameter-varying trim points $\boldsymbol{x}_{tr}(k_1)$, $\boldsymbol{u}_{tr}(k_1)$, and $\boldsymbol{w}_{tr} = \boldsymbol{0}$ yields the continuous-time LPV state–space system

$$\begin{bmatrix} \dot{\bar{\boldsymbol{x}}}(t) \\ \bar{\boldsymbol{z}}(t) \\ \bar{\boldsymbol{y}}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^c(k_1) & \boldsymbol{B}_1^{cw}(k_1) & \boldsymbol{B}_2^c(k_1) \\ \boldsymbol{C}_1^c(k_1) & \boldsymbol{D}_{11}^{cw}(k_1) & \boldsymbol{D}_{12}^c(k_1) \\ \boldsymbol{C}_2^c(k_1) & \boldsymbol{D}_{21}^{cw}(k_1) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{x}}(t) \\ \boldsymbol{w}(t) \\ \bar{\boldsymbol{u}}(t) \end{bmatrix}, \tag{13}$$

where t is continuous time, $\bar{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_{tr}$, $\bar{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_{tr}$, $\bar{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{y}_{tr}$, and $\bar{\boldsymbol{x}}(0) = \boldsymbol{0}$. The system matrices are calculated symbolically and are given by

$$\begin{aligned} \boldsymbol{A}^c(k_1) &= \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \quad \boldsymbol{B}_1^{cw}(k_1) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \\ \boldsymbol{B}_2^c(k_1) &= \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \quad \boldsymbol{C}_1^c(k_1) = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \\ \boldsymbol{D}_{11}^{cw}(k_1) &= \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{w}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \quad \boldsymbol{D}_{12}^c(k_1) = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \\ \boldsymbol{C}_2^c(k_1) &= \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \quad \boldsymbol{D}_{21}^{cw}(k_1) = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{w}}\right|_{(\boldsymbol{x}_{tr},\boldsymbol{w}_{tr},\boldsymbol{u}_{tr})}, \end{aligned}$$

where $\boldsymbol{x}_{tr}$ and $\boldsymbol{u}_{tr}$ are dependent on the parameter k_1. Note that $\boldsymbol{p}_e = \bar{\boldsymbol{p}}_e$. As the thrust model is based on lookup tables, it is linearized prior to the symbolic Jacobian calculations using the small-perturbation method, yielding the linear model $T = T_{\delta_T}\delta_T + T_{V_a}V_a$, where T_{δ_T} and T_{V_a} are constants.

The matrices in (13) have nonlinear dependence on the parameter k_1 due to the existence of trigonometric functions within the equations of motion (1), (6), and (7). Since $\phi_{e_{tr}}(k_1)$ is bounded as a result of k_1 being bounded, the zero-centered trigonometric functions of $\phi_{e_{tr}}(k_1)$ are approximated by the low-order Taylor series representations

$$\sin\phi_{e_{tr}}(k_1) \approx \phi_{e_{tr}}(k_1) \quad \text{and} \quad \cos\phi_{e_{tr}}(k_1) \approx 1 - \frac{1}{2}\phi_{e_{tr}}(k_1)^2. \tag{14}$$

Similar functions are defined for $\theta_{e_{tr}}(k_1)$. Substituting these functions into the matrices in (13) ensures that the continuous-time LPV system is polynomially dependent on k_1.

In order to achieve robust performance in the midst of disturbances, a weighting matrix based on the expected worst-case atmospheric disturbances and three times the sensor noise standard deviations is defined as $\boldsymbol{W}_w = \text{diag}(3\boldsymbol{I}_3, 0.026\boldsymbol{I}_3, 0.03\boldsymbol{I}_3, 6, 6\boldsymbol{I}_3)$. The disturbance input

matrices are then redefined as

$$\boldsymbol{B}_1^c(k_1) = \boldsymbol{W}_w \boldsymbol{B}_1^{cw}(k_1), \quad \boldsymbol{D}_{11}^c(k_1) = \boldsymbol{W}_w \boldsymbol{D}_{11}^{cw}(k_1), \quad \text{and}$$
$$\boldsymbol{D}_{21}^c(k_1) = \boldsymbol{W}_w \boldsymbol{D}_{21}^{cw}(k_1).$$

To enable real-time implementation of the path-following controllers on the small UAS platform, discrete-time controllers are designed with a sampling time of 0.04 s. Therefore, the continuous-time LPV model is discretized using the Euler method with a sampling time of $\tau = 0.04$ as

$$\boldsymbol{A}(k_1) = (\boldsymbol{I} + \tau \boldsymbol{A}^c(k_1)), \quad \boldsymbol{B}_i(k_1) = \tau \boldsymbol{B}_i^c(k_1), \quad \boldsymbol{C}_i(k_1) = \boldsymbol{C}_i^c(k_1),$$
$$\boldsymbol{D}_{i1}(k_1) = \boldsymbol{D}_{i1}^c(k_1), \quad \boldsymbol{D}_{12}(k_1) = \boldsymbol{D}_{12}^c(k_1),$$

for $i = 1, 2$. Euler discretization is used to maintain minimum order parameter dependence on k_1. The discrete-time LPV system can then be expressed as

$$\begin{bmatrix} \bar{\boldsymbol{x}}_{k+1} \\ \bar{\boldsymbol{z}}_k \\ \bar{\boldsymbol{y}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}(k_1) & \boldsymbol{B}_1(k_1) & \boldsymbol{B}_2(k_1) \\ \boldsymbol{C}_1(k_1) & \boldsymbol{D}_{11}(k_1) & \boldsymbol{D}_{12}(k_1) \\ \boldsymbol{C}_2(k_1) & \boldsymbol{D}_{21}(k_1) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{x}}_k \\ \boldsymbol{w}_k \\ \bar{\boldsymbol{u}}_k \end{bmatrix}, \tag{15}$$

where $\bar{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}(k\tau)$, $\bar{\boldsymbol{u}}_k = \bar{\boldsymbol{u}}(k\tau)$, $\bar{\boldsymbol{y}}_k = \bar{\boldsymbol{y}}(k\tau)$, $\bar{\boldsymbol{z}}_k = \bar{\boldsymbol{z}}(k\tau)$, $\boldsymbol{w}_k = \boldsymbol{w}(k\tau)$, and $\bar{\boldsymbol{x}}_0 = \boldsymbol{0}$.The scheduling parameter k_1 takes values in the interval $[-0.0141, 0.0141]$. However, the LPV synthesis result used in this paper requires that the scheduling parameter take values in the interval $[-1, 1]$. To facilitate the application of the synthesis result, a new scaled parameter $\bar{k}_1 = k_1/0.0141$ is used. Then, replacing the parameter k_1 in the system equation (15) with $0.0141\bar{k}_1$ leads to an LPV model with a scheduling parameter $\bar{k}_1$ such that $|\bar{k}_1| \leq 1$. Since the reformulated system (15) has polynomial dependence on the parameter $\bar{k}_1$, it can be equivalently represented by the LFR shown in Fig. 5, and defined as

$$\begin{bmatrix} \bar{\boldsymbol{x}}_{k+1} \\ \boldsymbol{\varphi}_k \\ \bar{\boldsymbol{z}}_k \\ \bar{\boldsymbol{y}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{ss} & \boldsymbol{A}_{sp} & \boldsymbol{B}_{1s} & \boldsymbol{B}_{2s} \\ \boldsymbol{A}_{ps} & \boldsymbol{A}_{pp} & \boldsymbol{B}_{1p} & \boldsymbol{B}_{2p} \\ \boldsymbol{C}_{1s} & \boldsymbol{C}_{1p} & \boldsymbol{D}_{11} & \boldsymbol{D}_{12} \\ \boldsymbol{C}_{2s} & \boldsymbol{C}_{2p} & \boldsymbol{D}_{21} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{x}}_k \\ \boldsymbol{\vartheta}_k \\ \boldsymbol{w}_k \\ \bar{\boldsymbol{u}}_k \end{bmatrix}, \tag{16}$$

where $\boldsymbol{\vartheta}_k = \bar{k}_1 \boldsymbol{\varphi}_k$, $\bar{\boldsymbol{x}}_0 = \boldsymbol{0}$, $\boldsymbol{\varphi}_k \in \mathbb{R}^{n_\delta}$, and $\boldsymbol{\vartheta}_k \in \mathbb{R}^{n_\delta}$. It is noted that the number of system states, n, is 18.

4.2. LPV $\mathcal{H}_\infty$ Controller

A discrete-time LPV controller with a sampling time of 0.04 s is synthesized for the LPV model (16). A parameter-independent Lyapunov approach is adopted to design the LPV controller based on Farhood (2012) and Packard (1994). The controller is designed to attenuate the effect of the exogenous disturbances $\boldsymbol{w}$ on the performance output $\bar{\boldsymbol{z}}$ by minimizing the ℓ_2-induced norm of the input–output mapping $\boldsymbol{w} \mapsto \bar{\boldsymbol{z}}$ for all permissible values of the scheduling parameter $\bar{k}_1$. In other words, a controller is sought which minimizes the ℓ_2-gain performance level γ such that

$$\|\boldsymbol{w} \mapsto \bar{\boldsymbol{z}}\|_{\ell_2 \to \ell_2} = \sup_{\|\boldsymbol{w}\|_{\ell_2} \neq 0} \frac{\|\bar{\boldsymbol{z}}\|_{\ell_2}}{\|\boldsymbol{w}\|_{\ell_2}} < \gamma \tag{17}$$

for all permissible values of $\bar{k}_1 \in [-1, 1]$. The procedure used for constructing the controller goes as follows. First, the following semi-definite program (SDP) is solved for $\boldsymbol{R}_s$, $\boldsymbol{R}_p$, $\boldsymbol{S}_s$, $\boldsymbol{S}_p$, and γ:

$$\begin{aligned} \text{minimize} \quad & : \quad \gamma^2 \\ \text{subject to} \quad & : \quad \boldsymbol{F}_s^T \boldsymbol{R}_s \boldsymbol{F}_s + \boldsymbol{F}_p^T \boldsymbol{R}_p \boldsymbol{F}_p - \boldsymbol{V}_{1s}^T \boldsymbol{R}_s \boldsymbol{V}_{1s} - \boldsymbol{V}_{1p}^T \boldsymbol{R}_p \boldsymbol{V}_{1p} \\ & \qquad + \boldsymbol{M}^T \boldsymbol{M} - \gamma^2 \boldsymbol{V}_2^T \boldsymbol{V}_2 \prec \boldsymbol{0}, \\ & \begin{bmatrix} \boldsymbol{W}_s^T \boldsymbol{S}_s \boldsymbol{W}_s + \boldsymbol{W}_p^T \boldsymbol{S}_p \boldsymbol{W}_p - \boldsymbol{U}_{1s}^T \boldsymbol{S}_s \boldsymbol{U}_{1s} - \boldsymbol{U}_{1p}^T \boldsymbol{S}_p \boldsymbol{U}_{1p} - \boldsymbol{U}_2^T \boldsymbol{U}_2 & \boldsymbol{L}^T \\ \boldsymbol{L} & -\gamma^2 \boldsymbol{I} \end{bmatrix} \prec \boldsymbol{0}, \\ & \begin{bmatrix} \boldsymbol{R}_s & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{S}_s \end{bmatrix} \succ \boldsymbol{0}, \quad \begin{bmatrix} \boldsymbol{R}_p & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{S}_p \end{bmatrix} \succ \boldsymbol{0}, \end{aligned}$$

where

$$\text{Im}\begin{bmatrix} \boldsymbol{V}_{1s}^T & \boldsymbol{V}_{1p}^T & \boldsymbol{V}_2^T \end{bmatrix}^T = \text{Ker}\begin{bmatrix} \boldsymbol{B}_{2s}^T & \boldsymbol{B}_{2p}^T & \boldsymbol{D}_{12}^T \end{bmatrix},$$
$$\begin{bmatrix} \boldsymbol{V}_{1s}^T & \boldsymbol{V}_{1p}^T & \boldsymbol{V}_2^T \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_{1s}^T & \boldsymbol{V}_{1p}^T & \boldsymbol{V}_2^T \end{bmatrix}^T = \boldsymbol{I},$$
$$\text{Im}\begin{bmatrix} \boldsymbol{U}_{1s}^T & \boldsymbol{U}_{1p}^T & \boldsymbol{U}_2^T \end{bmatrix}^T = \text{Ker}\begin{bmatrix} \boldsymbol{C}_{2s} & \boldsymbol{C}_{2p} & \boldsymbol{D}_{21} \end{bmatrix},$$
$$\begin{bmatrix} \boldsymbol{U}_{1s}^T & \boldsymbol{U}_{1p}^T & \boldsymbol{U}_2^T \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_{1s}^T & \boldsymbol{U}_{1p}^T & \boldsymbol{U}_2^T \end{bmatrix}^T = \boldsymbol{I},$$
$$\boldsymbol{F}_s = \boldsymbol{A}_{ss}^T \boldsymbol{V}_{1s} + \boldsymbol{A}_{ps}^T \boldsymbol{V}_{1p} + \boldsymbol{C}_{1s}^T \boldsymbol{V}_2, \quad \boldsymbol{F}_p = \boldsymbol{A}_{sp}^T \boldsymbol{V}_{1s} + \boldsymbol{A}_{pp}^T \boldsymbol{V}_{1p} + \boldsymbol{C}_{1p}^T \boldsymbol{V}_2,$$
$$\boldsymbol{W}_s = \boldsymbol{A}_{ss} \boldsymbol{U}_{1s} + \boldsymbol{A}_{sp} \boldsymbol{U}_{1p} + \boldsymbol{B}_{1s} \boldsymbol{U}_2, \quad \boldsymbol{W}_p = \boldsymbol{A}_{ps} \boldsymbol{U}_{1s} + \boldsymbol{A}_{pp} \boldsymbol{U}_{1p} + \boldsymbol{B}_{1p} \boldsymbol{U}_2,$$
$$\boldsymbol{M} = \boldsymbol{B}_{1s}^T \boldsymbol{V}_{1s} + \boldsymbol{B}_{1p}^T \boldsymbol{V}_{1p} + \boldsymbol{D}_{11}^T \boldsymbol{V}_2,$$
$$\boldsymbol{L} = \boldsymbol{C}_{1s} \boldsymbol{U}_{1s} + \boldsymbol{C}_{1p} \boldsymbol{U}_{1p} + \boldsymbol{D}_{11} \boldsymbol{U}_2,$$

with Im and Ker denoting the image and kernel, respectively. Define, for convenience, $\bar{n} = n + n_\delta$,

$$\boldsymbol{E}_s = (\boldsymbol{R}_s - \boldsymbol{S}_s^{-1})^{1/2}, \quad \boldsymbol{E}_p = (\boldsymbol{R}_p - \boldsymbol{S}_p^{-1})^{1/2}, \quad \boldsymbol{E} = \text{diag}(\boldsymbol{E}_s, \boldsymbol{E}_p),$$
$$\boldsymbol{R} = \text{diag}(\boldsymbol{R}_s, \boldsymbol{R}_p),$$

$$\boldsymbol{S} = \text{diag}(\boldsymbol{S}_s, \boldsymbol{S}_p), \quad \boldsymbol{B}_i = \begin{bmatrix} \boldsymbol{B}_{is}^T & \boldsymbol{B}_{ip}^T \end{bmatrix}^T \text{ and } \boldsymbol{C}_i = \begin{bmatrix} \boldsymbol{C}_{is} & \boldsymbol{C}_{ip} \end{bmatrix} \text{ for } i = 1, 2.$$

Next, solve the following SDP for $\boldsymbol{J}$:

$$\boldsymbol{H} + \boldsymbol{P}^T \boldsymbol{J} \boldsymbol{Q} + \boldsymbol{Q}^T \boldsymbol{J}^T \boldsymbol{P} \prec \boldsymbol{0},$$

where

$$\boldsymbol{H} = \begin{bmatrix} -\boldsymbol{Y} & \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{A}}^T & -\boldsymbol{X} & \boldsymbol{\mathcal{C}} \\ \boldsymbol{\mathcal{B}}^T & \boldsymbol{\mathcal{C}}^T & \boldsymbol{\mathcal{D}} \end{bmatrix}, \quad \boldsymbol{P} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0}_{\bar{n} \times (2\bar{n} + n_w)} & \boldsymbol{0} \\ \boldsymbol{B}_2^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}_{12}^T/\gamma \end{bmatrix},$$
$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{0}_{\bar{n} \times 2\bar{n}} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}_{\bar{n} \times n_z} \\ \boldsymbol{0} & \boldsymbol{C}_2 & \boldsymbol{0} & \boldsymbol{D}_{21} & \boldsymbol{0} \end{bmatrix},$$

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{E} \\ \boldsymbol{E}^T & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{S} & -\boldsymbol{S}\boldsymbol{E} \\ -\boldsymbol{E}^T \boldsymbol{S}^T & \boldsymbol{I} + \boldsymbol{E}^T \boldsymbol{S} \boldsymbol{E} \end{bmatrix}, \quad \boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{A}_{ss} & \boldsymbol{A}_{sp} & \boldsymbol{0} \\ \boldsymbol{A}_{ps} & \boldsymbol{A}_{pp} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}_{\bar{n} \times \bar{n}} \end{bmatrix},$$
$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}_{\bar{n} \times n_z} \end{bmatrix}, \quad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{C}_1^T/\gamma \\ \boldsymbol{0} & \boldsymbol{0}_{\bar{n} \times n_z} \end{bmatrix}, \text{ and } \boldsymbol{\mathcal{D}} = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{D}_{11}^T/\gamma \\ \boldsymbol{D}_{11}/\gamma & -\boldsymbol{I} \end{bmatrix}.$$

Finally, the controller can be constructed from $\boldsymbol{J}$ as an LFR with the state–space realization

$$\begin{bmatrix} \boldsymbol{x}_{k+1}^K \\ \boldsymbol{\varphi}_k^K \\ \bar{\boldsymbol{u}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{ss}^K & \boldsymbol{A}_{sp}^K & \boldsymbol{B}_s^K \\ \boldsymbol{A}_{ps}^K & \boldsymbol{A}_{pp}^K & \boldsymbol{B}_p^K \\ \boldsymbol{C}_s^K & \boldsymbol{C}_p^K & \boldsymbol{D}^K \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k^K \\ \boldsymbol{\vartheta}_k^K \\ \bar{\boldsymbol{y}}_k \end{bmatrix}, \quad \boldsymbol{\vartheta}_k^K = \bar{k}_1 \boldsymbol{\varphi}_k^K,$$
$$\text{where } \boldsymbol{J} = \begin{bmatrix} \boldsymbol{A}_{ss}^K & \boldsymbol{A}_{sp}^K & \boldsymbol{B}_s^K \\ \boldsymbol{A}_{ps}^K & \boldsymbol{A}_{pp}^K & \boldsymbol{B}_p^K \\ \boldsymbol{C}_s^K & \boldsymbol{C}_p^K & \boldsymbol{D}^K \end{bmatrix},$$

$\boldsymbol{x}_k^K \in \mathbb{R}^n$, $\boldsymbol{\vartheta}_k^K \in \mathbb{R}^{n_\delta}$, and $\boldsymbol{x}_0^K = \boldsymbol{0}$. The state–space realization of the controller after closing the LFR is given by

$$\begin{bmatrix} \boldsymbol{x}_{k+1}^K \\ \bar{\boldsymbol{u}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{cl}^K(\bar{k}_1) & \boldsymbol{B}_{cl}^K(\bar{k}_1) \\ \boldsymbol{C}_{cl}^K(\bar{k}_1) & \boldsymbol{D}_{cl}^K(\bar{k}_1) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k^K \\ \bar{\boldsymbol{y}}_k \end{bmatrix},$$

where

$$\boldsymbol{A}_{cl}^K(\bar{k}_1) = \boldsymbol{A}_{ss}^K + \boldsymbol{A}_{sp}^K \boldsymbol{\Delta} \boldsymbol{A}_{ps}^K, \; \boldsymbol{B}_{cl}^K(\bar{k}_1) = \boldsymbol{B}_s^K + \boldsymbol{A}_{sp}^K \boldsymbol{\Delta} \boldsymbol{B}_p^K,$$
$$\boldsymbol{C}_{cl}^K(\bar{k}_1) = \boldsymbol{C}_s^K + \boldsymbol{C}_p^K \boldsymbol{\Delta} \boldsymbol{A}_{ps}^K,$$
$$\boldsymbol{D}_{cl}^K(\bar{k}_1) = \boldsymbol{D}^K + \boldsymbol{C}_p^K \boldsymbol{\Delta} \boldsymbol{B}_p^K, \quad \text{and} \quad \boldsymbol{\Delta} = \bar{k}_1 (\boldsymbol{I} - \bar{k}_1 \boldsymbol{A}_{pp}^K)^{-1}.$$

The performance output for the controller is chosen as

$$\bar{\boldsymbol{z}} = [\; 0.1\bar{V}_a,\; 3 \times 10^{-3}\bar{p},\; 2.2\bar{p},\; 1.5\bar{r},\; 0.5\bar{\phi}_e,\; 0.85\bar{\theta}_e,\; 0.45\bar{\psi}_e,\; 0.0255x_e,$$
$$0.2y_e,\; 0.05z_e, 0.8\bar{\delta}_E,\; 1.1\bar{\delta}_{A_{\text{dyn}}},\; 4.0\bar{\delta}_{R_{\text{dyn}}},\; 3.4\bar{\delta}_T\;]^T,$$

where $\bar{\delta}_{A_{\text{dyn}}}$ and $\bar{\delta}_{R_{\text{dyn}}}$ are the aileron and rudder deflections passed through first-order dynamic filters, respectively. The choice of the performance output is driven by the requirement to minimize the position

error between the UAS and the virtual vehicle and to have minimum control actuation. In addition to penalizing the errors in the position and attitude angles, it is found that penalizing the sideslip angle, roll rate, and yaw rate decreases the likelihood of rudder saturation during tighter turns. To minimize the high-frequency oscillations observed in the aileron and rudder deflections, the high-frequency components are more heavily penalized by passing them through high-pass filters. First-order discrete-time high-pass filters with a cut-off frequency of 1.5 Hz and 1 Hz are used on the aileron and rudder deflections, respectively. The introduction of the dynamic filters increases the number of system states in (16) by two. Since incorporating dynamic weights results in additional system states and hence more controller states, in the control design it is preferred to restrict the use of dynamic weights unless deemed necessary. With this said, the utility of using dynamic penalty weights on states, as well as functions of states and disturbances, was actually explored during control design, but it was found that any improvement in performance achieved was not significant and did not justify the increase in problem size.

During the control design process, it is found that approximating q_{tr}, ϕ_{tr}, θ_{tr}, w_{tr}, and $\delta_{E_{tr}}$ as parameter-varying trims did not improve the controller performance and simply increased the size of the LFR. Therefore, the four trim states and the trim elevator input are fixed at the values corresponding to $k_1 = 0$ resulting in $n_\delta = 10$. All computations are carried out on a Dell Precision T3500 Desktop running 64-bit Windows 7, with an Intel Xeon W3550 Quad Core processor and 6 GB of RAM. SDPs in the controller synthesis problem are solved in `MATLAB` 2014a using the `YALMIP` toolbox (Löfberg, 2004) with `SeDuMi` (Sturm, 1999) as the solver, and the SDP to obtain J, that is, the controller state-space matrices, is solved using the `MATLAB` function `basiclmi`. The resulting controller synthesis problem is solved in 12.5 s, and the optimal value of γ is found to be $\gamma_{\min} = 2.61$, which is relaxed to $\gamma = 5.21$ to obtain satisfactory robust performance, and the synthesis problem is re-solved. The resulting LPV controller has 20 system states and 10 states pertaining to the parameter $\bar{k}_1$.

Balanced truncation model reduction of the LPV $\mathcal{H}_\infty$ Controller

In view of the onboard implementation of the LPV controller on the UAS platform and its real-time execution, the higher number of states of the LPV controller poses a challenge. Therefore, it is necessary to reduce the number of states of the LPV controller without losing its stability and performance guarantees by an appropriate model reduction technique. The Balanced Truncation (BT) model reduction method for systems represented by an LFR, described in Beck, Doyle, and Glover (1996) and Beck (2006), is a suitable method as it provides a guaranteed upper bound on the error between the original and the reduced system.

With the notion of strong stability as defined in Beck (2006), the LPV $\mathcal{H}_\infty$ controller, denoted by k, with state–space matrices given by $\boldsymbol{J}$ is found to be strongly stable, thereby making it possible to apply the BT model reduction method. Before presenting the model reduction procedure, the matrices $\boldsymbol{A}^K$, $\boldsymbol{B}^K$, and $\boldsymbol{C}^K$ are defined from $\boldsymbol{J}$ as

$$\boldsymbol{A}^K = \begin{bmatrix} \boldsymbol{A}^K_{ss} & \boldsymbol{A}^K_{sp} \\ \boldsymbol{A}^K_{ps} & \boldsymbol{A}^K_{pp} \end{bmatrix}, \quad \boldsymbol{B}^K = \begin{bmatrix} \boldsymbol{B}^K_{s} \\ \boldsymbol{B}^K_{p} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{C}^K = \begin{bmatrix} \boldsymbol{C}^K_{s} & \boldsymbol{C}^K_{p} \end{bmatrix}.$$

The controllability gramian, $\boldsymbol{X}$, and the observability gramian, $\boldsymbol{Y}$, are obtained by solving the following Lyapunov inequalities:

$$\boldsymbol{A}^K \boldsymbol{X} (\boldsymbol{A}^K)^T - \boldsymbol{X} + \boldsymbol{B}^K (\boldsymbol{B}^K)^T \prec \boldsymbol{0} \quad \text{and}$$
$$(\boldsymbol{A}^K)^T \boldsymbol{Y} \boldsymbol{A}^K - \boldsymbol{Y} + (\boldsymbol{C}^K)^T \boldsymbol{C}^K \prec \boldsymbol{0}.$$

The gramians $\boldsymbol{X}$ and $\boldsymbol{Y}$ are positive definite block-diagonal matrices that are structured according to the partitions in $\boldsymbol{A}^K$. In particular, solutions $\boldsymbol{X}$ and $\boldsymbol{Y}$ with minimum trace are solicited to have a reasonable error bound.

Using the solutions $\boldsymbol{X}$ and $\boldsymbol{Y}$, a balancing transformation $\boldsymbol{T}$ and a diagonal gramian $\boldsymbol{\Sigma}$ are constructed as outlined in Section 6.9 of Gugercin and Antoulas (2004). The balancing transformation $\boldsymbol{T}$ is then used to construct a balanced system with the following state–space matrices:

$$\hat{\boldsymbol{A}}^K = \boldsymbol{T} \boldsymbol{A}^K \boldsymbol{T}^{-1}, \quad \hat{\boldsymbol{B}}^K = \boldsymbol{T} \boldsymbol{B}^K, \quad \hat{\boldsymbol{C}}^K = \boldsymbol{C}^K \boldsymbol{T}^{-1}, \quad \hat{\boldsymbol{D}}^K = \boldsymbol{D}^K.$$

Note that $\boldsymbol{T}$ and $\boldsymbol{\Sigma}$ are also structured according to the partitions in $\boldsymbol{A}^K$. The diagonal gramian $\boldsymbol{\Sigma}$ is given by $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\Sigma_1}, \boldsymbol{\Sigma_2})$, where

$$\boldsymbol{\Sigma_1} = \text{diag}(\sigma_1 \boldsymbol{I}_{d_1}, \ldots, \sigma_{j_1} \boldsymbol{I}_{d_{j_1}}, \ldots, \sigma_L \boldsymbol{I}_{d_L}), \quad \sum_{i=1}^{L} d_i = n,$$

$$\boldsymbol{\Sigma_2} = \text{diag}(\sigma_{L+1} \boldsymbol{I}_{d_{L+1}}, \ldots, \sigma_{j_2} \boldsymbol{I}_{d_{j_2}}, \ldots, \sigma_M \boldsymbol{I}_{d_M}), \quad \sum_{i=L+1}^{M} d_i = n_\delta,$$

$$\sigma_1 > \sigma_2 > \cdots > \sigma_L > 0, \quad \sigma_{L+1} > \sigma_{L+2} > \cdots > \sigma_M > 0,$$

and d_i is the multiplicity of σ_i. The singular values less than σ_{j_1} and σ_{j_2} are truncated from $\boldsymbol{\Sigma_1}$ and $\boldsymbol{\Sigma_2}$, respectively, which results in the following bound on the error between the original system, $\boldsymbol{K}$, and the reduced-order system, $\boldsymbol{K_r}$, for all permissible values of the scheduling parameter:

$$\|\boldsymbol{K} - \boldsymbol{K_r}\|_{\ell_2 \to \ell_2} \leq 2(\sigma_{j_1+1} + \cdots + \sigma_L) + 2(\sigma_{j_2+1} + \cdots + \sigma_M).$$

The reduced-order system is obtained from the balanced system by removing the rows and columns corresponding to the truncated states from the system matrices. The singular values comprising $\boldsymbol{\Sigma}$ and the truncated singular values for the LPV controller $\boldsymbol{K}$ are shown in Fig. 7(a). Using 0.05 as a threshold, four system states and three parameter states are truncated with an error bound of 0.253 and a relative error bound of 6.5%. To obtain a tighter error bound, a heuristic based on the 1-norm is used to obtain a larger number of repeated singular values, see Abou Jaoude and Farhood (2017) for more details. Since repeated singular values are counted only once in the computation of the error bound, this heuristic helps in reducing the error bound. Using the heuristic, the error bound for truncating the same number of states is reduced to 0.093 and the relative error bound to 2.3%. Fig. 7(b) shows the singular values and the truncated singular values using the heuristic. The reduced-order controller has 23 states consisting of 16 system states and 7 parameter states. The performance of the reduced-order controller is comparable to that of the original LPV controller, and the reduction in position tracking performance is less than 0.2%.

4.3. LTI $\mathcal{H}_\infty$ Controller

A standard $\mathcal{H}_\infty$ controller, henceforth referred to as the LTI controller, is designed for the LTI plant obtained from (15) by taking $k_1 = 0$. The synthesis procedure for designing a standard $\mathcal{H}_\infty$ controller is described in detail in Gahinet and Apkarian (1994), which is also presented here for the sake of completeness. Firstly, the following SDP is solved for γ, $\boldsymbol{R}$, and $\boldsymbol{S}$:

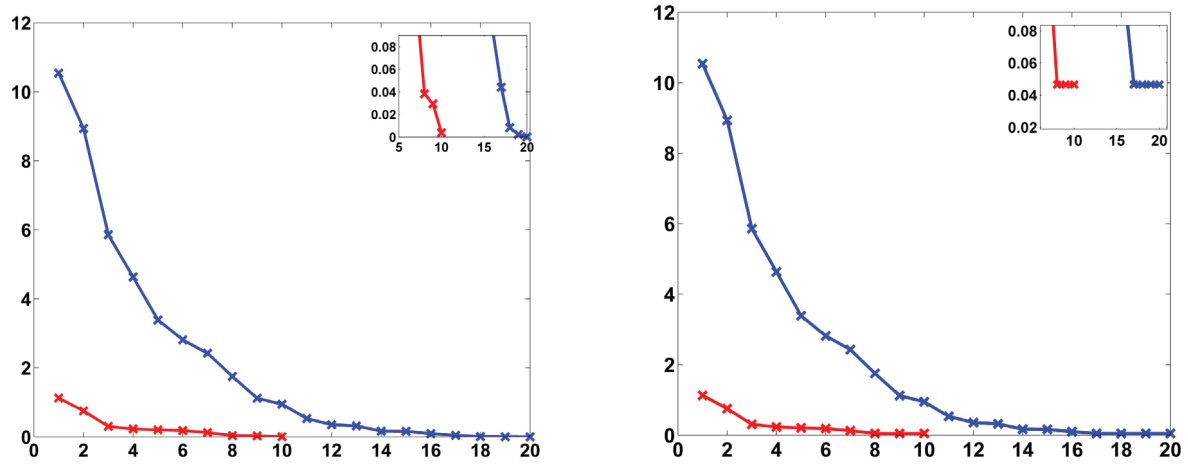
$$\begin{aligned} \text{minimize} \ : \ & \gamma^2 \\ \text{subject to} \ : \ & \boldsymbol{F}^T \boldsymbol{R} \boldsymbol{F} - \boldsymbol{V}_1^T \boldsymbol{R} \boldsymbol{V}_1 + \boldsymbol{M}^T \boldsymbol{M} - \gamma^2 \boldsymbol{V}_2^T \boldsymbol{V}_2 \prec \boldsymbol{0}, \\ & \begin{bmatrix} \boldsymbol{W}^T \boldsymbol{S} \boldsymbol{W} - \boldsymbol{U}_1^T \boldsymbol{S} \boldsymbol{U}_1 - \boldsymbol{U}_2^T \boldsymbol{U}_2 & \boldsymbol{L}^T \\ \boldsymbol{L} & -\gamma^2 \boldsymbol{I}_{n_z} \end{bmatrix} \prec \boldsymbol{0}, \\ & \begin{bmatrix} \boldsymbol{R} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{S} \end{bmatrix} \succeq \boldsymbol{0}, \end{aligned}$$

where

$$\boldsymbol{F} = \boldsymbol{A}^T \boldsymbol{V}_1 + \boldsymbol{C}_1^T \boldsymbol{V}_2, \quad \boldsymbol{M} = \boldsymbol{B}_1^T \boldsymbol{V}_1 + \boldsymbol{D}_{11}^T \boldsymbol{V}_2,$$
$$\boldsymbol{W} = \boldsymbol{A} \boldsymbol{U}_1 + \boldsymbol{B}_1 \boldsymbol{U}_2, \quad \boldsymbol{L} = \boldsymbol{C}_1 \boldsymbol{U}_1 + \boldsymbol{D}_{11} \boldsymbol{U}_2,$$
$$\text{Im} \begin{bmatrix} \boldsymbol{V}_1^T & \boldsymbol{V}_2^T \end{bmatrix}^T = \text{Ker} \begin{bmatrix} \boldsymbol{B}_2^T & \boldsymbol{D}_{12}^T \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{V}_1^T & \boldsymbol{V}_2^T \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1^T & \boldsymbol{V}_2^T \end{bmatrix}^T = \boldsymbol{I},$$
$$\text{Im} \begin{bmatrix} \boldsymbol{U}_1^T & \boldsymbol{U}_2^T \end{bmatrix}^T = \text{Ker} \begin{bmatrix} \boldsymbol{C}_2 & \boldsymbol{D}_{21} \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{U}_1^T & \boldsymbol{U}_2^T \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1^T & \boldsymbol{U}_2^T \end{bmatrix}^T = \boldsymbol{I}.$$

Next, using the solutions $\boldsymbol{R}$ and $\boldsymbol{S}$, the following SDP is solved for $\boldsymbol{J}$:

$$\boldsymbol{H} + \boldsymbol{P}^T \boldsymbol{J} \boldsymbol{Q} + \boldsymbol{Q}^T \boldsymbol{J}^T \boldsymbol{P} \prec \boldsymbol{0},$$



(a) Singular values of $\boldsymbol{\Sigma}$. (b) Singular values of $\boldsymbol{\Sigma}$ obtained using the heuristic.

Fig. 7. The singular values pertaining to $\boldsymbol{\Sigma}_1$ are shown in blue and the singular values pertaining to $\boldsymbol{\Sigma}_2$ are shown in red, and the truncated singular values are shown in the inset.

where

$$\boldsymbol{H} = \begin{bmatrix} -\boldsymbol{X}^{-1} & \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{A}}^T & -\boldsymbol{X} & \boldsymbol{C}_1 \\ \boldsymbol{\mathcal{B}}^T & \boldsymbol{C}_2 & \boldsymbol{\mathcal{D}} \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{S} & -\boldsymbol{S}\boldsymbol{M} \\ -\boldsymbol{M}^T\boldsymbol{S} & \boldsymbol{I} + \boldsymbol{M}^T\boldsymbol{S}\boldsymbol{M} \end{bmatrix},$$

$$\boldsymbol{X}^{-1} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{M} \\ \boldsymbol{M}^T & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{M} = (\boldsymbol{R} - \boldsymbol{S}^{-1})^{1/2},$$

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0}_{n\times 2n+n_w} & \boldsymbol{0} \\ \boldsymbol{B}_2^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}_{12}^T/\gamma \end{bmatrix},$$

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{0}_{n\times 2n} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}_{n\times n_z} \\ \boldsymbol{0} & \boldsymbol{C}_2 & \boldsymbol{0} & \boldsymbol{D}_{21} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}_{n\times n} \end{bmatrix},$$

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}_{n\times n_d} \end{bmatrix}, \quad \boldsymbol{C}_1 = \begin{bmatrix} \boldsymbol{C}_1^T/\gamma & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}_{n\times n_z} \end{bmatrix},$$

$$\boldsymbol{C}_2 = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0}_{n_d\times n} \\ \boldsymbol{C}_1/\gamma & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{\mathcal{D}} = \begin{bmatrix} -\boldsymbol{I}_{n_w} & \boldsymbol{D}_{11}^T/\gamma \\ \boldsymbol{D}_{11}/\gamma & -\boldsymbol{I}_{n_z} \end{bmatrix}.$$

Finally, the state–space model of the LTI controller can be constructed from $\boldsymbol{J}$ as follows:

$$\begin{bmatrix} \boldsymbol{x}_{k+1}^K \\ \bar{\boldsymbol{u}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^K & \boldsymbol{B}^K \\ \boldsymbol{C}^K & \boldsymbol{D}^K \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k^K \\ \bar{\boldsymbol{y}}_k \end{bmatrix}, \quad \text{where } \boldsymbol{J} = \begin{bmatrix} \boldsymbol{A}^K & \boldsymbol{B}^K \\ \boldsymbol{C}^K & \boldsymbol{D}^K \end{bmatrix} \text{ and } \boldsymbol{x}_0^K = \boldsymbol{0}.$$

The performance output for the LTI controller in Fig. 4 is chosen as

$$\bar{\boldsymbol{z}} = [\ 0.3\bar{V}_a,\ 0.15\bar{\beta},\ 0.25\bar{\phi}_e,\ 0.85\bar{\theta}_e,\ 0.45\bar{\psi}_e,\ 0.0255x_e,\ 0.08y_e,\ 0.153z_e, 0.9\bar{\delta}_E,\ 0.09\bar{\delta}_A,\ 1.1\bar{\delta}_R,\ 3.0\bar{\delta}_T\ ]^T.$$

The controller synthesis problem is solved in 9.6 s and an optimal value $\gamma_{\min} = 0.91$ is found. The value of γ is then relaxed to 1.82 to increase robustness, and the control synthesis problem is re-solved.

4.4. Rate-tracking controller

The rate-tracking (RT) controller consists of a nonlinear outer-loop control law based on Kaminer et al. (2010), which generates pitch and yaw rate commands, and an inner-loop $\mathcal{H}_\infty$ controller which tracks the commands generated by the nonlinear outer-loop control law. The nonlinear outer-loop control law is designed via backstepping, whereby the pitch and yaw rates, q and r, play the role of virtual control inputs. These commands are defined as

$$\begin{bmatrix} q_c \\ r_c \end{bmatrix} = \boldsymbol{Q}_c^{-1}(\theta_e, \phi_e) \left(\begin{bmatrix} \theta_c \\ \psi_c \end{bmatrix} - \boldsymbol{D}_c(\theta_e, \psi_e, \ell) \right), \tag{18}$$

with the following auxiliary definitions:

$$\boldsymbol{Q}_c(\theta_e, \phi_e) = \begin{bmatrix} \cos\phi_e & -\sin\phi_e \\ \dfrac{\sin\phi_e}{\cos\theta_e} & \dfrac{\cos\phi_e}{\cos\theta_e} \end{bmatrix},$$

$$\boldsymbol{D}_c(\theta_e, \psi_e, \ell) = \dot{\ell} \begin{bmatrix} k_2(\ell)\cos\psi_e \\ -k_1(\ell) + k_2(\ell)\tan\theta_e \sin\psi_e \end{bmatrix},$$

$$\theta_c = -K_2(\theta_e - \theta_a) + C_3 z_e V_a \frac{\sin\theta_e - \sin\theta_a}{\theta_e - \theta_a} + \dot{\theta}_a,$$

$$\psi_c = -K_3(\psi_e - \psi_a) + C_3 y_e V_a \cos\theta_e \frac{\sin\psi_e - \sin\psi_a}{\psi_e - \psi_a} + \dot{\psi}_a,$$

$$\theta_a = \sin^{-1}\frac{z_e}{|z_e| + d_1}, \quad \psi_a = \sin^{-1}\frac{-y_e}{|y_e| + d_2},$$

where d_1, d_2, K_2, K_3, and C_3 are positive constants. In Kaminer et al. (2010), the pitch and yaw rate commands, q_c and r_c, generated by the nonlinear control law are tracked by a Piccolo autopilot augmented by an $\mathcal{L}_1$ adaptive controller in the inner loop. In this work, the nonlinear control law (18) is used to generate q_c and r_c, which are then tracked by a standard $\mathcal{H}_\infty$ controller.

Instead of tracking the pitch and yaw rate commands given by (18) directly, the output of an ideal response model is tracked as in the example found in Doyle, Lenz, and Packard (1986). The pitch and yaw rate commands are passed through a second-order filter with $\omega_n = 12\,\text{rad/s}$ and $\zeta = 0.8$ to obtain the ideal pitch and yaw rate commands, which the inner-loop $\mathcal{H}_\infty$ controller seeks to track. The LTI plant for the inner-loop $\mathcal{H}_\infty$ controller is obtained in a manner similar to what is described in Sections 4.2 and 4.3. The state, measurement, and disturbance vectors are defined as $\boldsymbol{x} = [\boldsymbol{v}^T, \boldsymbol{\omega}^T, \phi_e, \theta_e, \boldsymbol{x}_a^T, \boldsymbol{x}_m^T]^T$, $\boldsymbol{w} = [\boldsymbol{v}_w^T, \boldsymbol{w}_m^T, \boldsymbol{w}_c^T]^T$, and $\boldsymbol{y} = [p,\ q - q_{\text{ideal}},\ r - r_{\text{ideal}},\ V_a,\ \phi_e,\ \theta_e]^T$, where $\boldsymbol{w}_m = [m_p,\ m_q,\ m_r,\ m_{V_a},\ m_\phi,\ m_\theta]^T$, $\boldsymbol{w}_c = [q_c,\ r_c]^T$. q_{ideal} and r_{ideal} are the outputs of the ideal response model, and $\boldsymbol{x}_m$ denotes the states corresponding to the two second-order filters of the ideal response model. The weighting matrix $\boldsymbol{W}_w$ is chosen as $\boldsymbol{W}_w = \text{diag}(3\boldsymbol{I}_3, 0.026\boldsymbol{I}_3, 6, 0.03\boldsymbol{I}_2, \boldsymbol{I}_2)$.

The synthesis procedure for the inner-loop $\mathcal{H}_\infty$ controller parallels the steps outlined in Section 4.3 and is therefore omitted here. The performance output for the controller is chosen as

$$\bar{\boldsymbol{z}} = [q - q_{\text{ideal}},\ 1.52(r - r_{\text{ideal}} + 1.57\bar{\delta}_A),\ 0.02\bar{V}_a,\ 0.6\bar{\alpha},\ 0.6\bar{\beta},\ \bar{\delta}_E,\ 0.84\bar{\delta}_A,\ 0.34\bar{\delta}_R,\ 0.1\bar{\delta}_T]^T.$$

The controller synthesis problem is solved in 3.9 s, with $\gamma_{\min} = 1.0$. The value of γ is relaxed to 2.0 to improve robustness, and the control synthesis problem is re-solved.

Table 3
Controller Parameters for the PID Path-Following Controller.

Parameter	Value	Parameter	Value	Parameter	Value
PTCH2SRV_TCONST	0.4	RLL2SRV_TCONST	0.5	YAW2SRV_INT	0.0
PTCH2SRV_P	1.85	RLL2SRV_P	0.4	YAW2SRV_DAMP	0.1
PTCH2SRV_D	0.02	RLL2SRV_D	0.02	YAW2SRV_SLIP	0.0
PTCH2SRV_I	0.06	RLL2SRV_I	0.04	YAW2SRV_RLL	1.0
PTCH2SRV_RLL	1.0	ARSPD_FBW_MIN	9.0	TECS_INTEG_GAIN	0.1
PTCH2SRV_RMA_DN	-25 deg	ARSPD_FBW_MAX	22.0	TECS_RLL_THR	10.0
PTCH2SRV_RMA_UP	20 deg	TECS_SPDWEIGHT	1.0	THR_SLEWRATE	100
NAVL1_PERIOD	20	NAVL1_DAMPING	0.75	TECS_THR_DAMP	0.5
THR_MAX	0.9	THR_MIN	0.1	TRIM_THROTTLE	0.45
TECS_CLIMB_MAX	5.0	TECS_SINK_MAX	5.0	TECS_TIME_CONST	4.0

4.5. PID path-following controller

The three controllers presented thus far are model-based controllers. In this subsection, a model-free path-following controller based on the controller used in the commercially available and widely used autopilot software ArduPilot is presented. This controller, henceforth referred to as the PID controller, is selected to compare the performance of the robust control methods presented in this paper with a widely used control method in the UAS community.

The PID controller has a cascaded architecture, where the inner-loop is composed of three PID controllers for each of the three control surfaces, and a total energy controller for the throttle. A nonlinear guidance logic based on Park et al. (2004) forms the outer-loop, and generates lateral acceleration commands depending on the speed of the aircraft and how far the aircraft is from the reference path. The guidance logic has only two modes, one for tracking a straight line path and the other for tracking a circular path. This limits the reference paths to straight lines, circles, and concatenations of straight line and circular paths. For more details on the PID controller, the interested reader is referred to ArduPilot (2016b). The PID controller is tuned based on closed-loop simulations with the nonlinear UAS model described in Section 2. The controller parameters are provided in Table 3, where the terminology used is the same as in ArduPilot (2016b).

5. Simulation environment

A rigorous MATLAB simulation environment is used to compare the performances of the controllers presented in the previous section. The simulation environment is designed to subject the small UAS to proportionally significant atmospheric disturbances while mimicking the implementation of the controller onboard the Telemaster UAS platform. The nonlinear rigid body equations of motion (1) are simulated using ODE23. The simulation environment includes the actuator dynamics and the second-order throttle dynamics described in Section 2.2. A one-step time delay is also included in simulations to mimic the autopilot system onboard the Telemaster UAS. Namely, measurements taken at the discrete time instant k are used to calculate the control input applied at time $k+1$. The LTI, LPV, and PID path-following controllers are executed at 25 Hz. While the inner-loop of the RT controller is executed at 25 Hz, the outer-loop is executed at a slower rate of 12.5 Hz to enable real-time implementation.

5.1. Simulated disturbances

The UAS is subjected to a 3.5 m/s steady wind in the North-East direction in addition to wind gusts modeled using the Dryden wind turbulence model (MIL-F-8785C) (Moorhouse & Woodcock, 1982). The UAS operates at altitudes under 1000 ft AGL, so corresponding low altitude scalings and wind intensities are used (Gage, 2003). The turbulence wind velocities are given by the following transfer functions:

$$\begin{aligned} H_u(s) &= \frac{u_{20}}{10h^{1/3}}\sqrt{\frac{2H}{\pi V_a h}}\,\frac{1}{1+\frac{H}{V_a h}s}, \\ H_v(s) &= \frac{u_{20}}{10h^{1/3}}\sqrt{\frac{H}{\pi V_a h}}\,\frac{1}{\left(1+\frac{H}{V_a h}s\right)^2}, \\ H_w(s) &= \frac{u_{20}}{10}\sqrt{\frac{H}{\pi V_a}}\,\frac{1}{\left(1+\frac{H}{V_a}s\right)^2}, \end{aligned} \tag{19}$$

where H is the current vehicle altitude in ft AGL, u_{20} is the average wind speed at 20 ft AGL in knots, and $h=(0.177+0.000823H)^{1.2}$. A value of $u_{20}=30$ knots is used in this work.

Sensor noise is simulated from a normal distribution with standard deviations of $\sigma_{V_a}=2\,\text{m/s}$, $\sigma_{p/q/r}=0.5\,\text{deg/s}$, $\sigma_{\phi/\theta/\psi}=0.57\,\text{deg}$, and $\sigma_{N/E/H}=2\,\text{m}$. The values for the standard deviations are obtained from sensor specifications and bench tests. It is noted that identical time histories of atmospheric disturbances and sensor noise are used with each controller investigated.

5.2. Reference paths

The reference paths chosen for comparing the different controllers are shown in Fig. 8. The lemniscate path has a smooth variation of k_1 and is generated by the functions

$$N_{\text{ref}}(\xi)=\frac{3\cos(\xi)}{k_{1_{\max}}(1+\sin(\xi)^2)} \quad \text{and} \quad E_{\text{ref}}(\xi)=\frac{3\sin(\xi)\cos(\xi)}{k_{1_{\max}}(1+\sin(\xi)^2)} \tag{20}$$

for $\xi\in[\pi/2,\ 5\pi/2]$, where $k_{1_{\max}}$ is a scaling parameter representing the maximum value of k_1 over the entire path. Since only planar paths are considered, $H_{\text{ref}}(\xi)=0$ for all $\xi\in[\pi/2,\ 5\pi/2]$. A circular path with a constant value of k_1 equal to $k_{1_{\max}}$ is generated by the functions

$$N_{\text{ref}}(\xi)=\frac{\cos(\xi)}{k_{1_{\max}}} \quad \text{and} \quad E_{\text{ref}}(\xi)=\frac{\sin(\xi)}{k_{1_{\max}}} \quad \text{for} \quad \xi\in[0,\ 2\pi]. \tag{21}$$

The reference paths shown in Fig. 8 correspond to $k_{1_{\max}}=0.0071$, $k_{1_{\max}}=0.009$, and $k_{1_{\max}}=0.0125$ for the moderate lemniscate, circular, and the tighter lemniscate paths, respectively.

5.3. Performance metrics

Several performance metrics are used in order to quantitatively compare the performances of the different controllers. The three criteria used are the mean path error, the control effort, and the average path time.

In order to accurately calculate the path error, the parameter interval of ξ is finely discretized into n_ξ points denoted as $\xi_1,\xi_2,\ldots,\xi_{n_\xi}$. The set of parameterized reference points R characterizing the reference path is then defined as

$$R=\left\{\boldsymbol{p}(\xi_i)\ \text{for}\ i=1,2,\ldots,n_\xi\right\}, \tag{22}$$

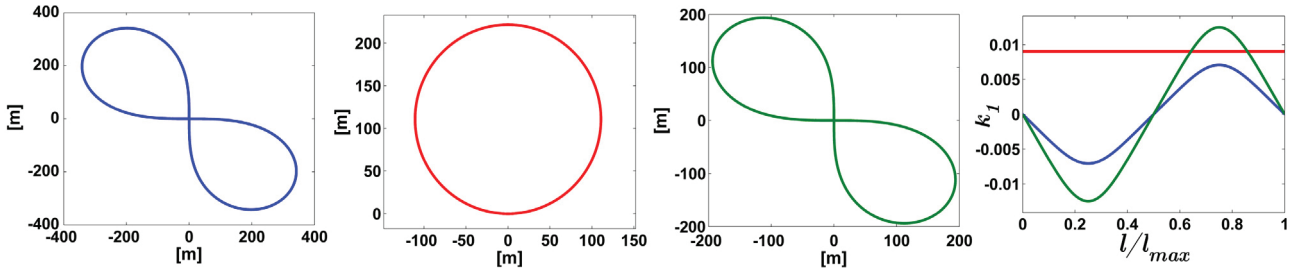


Fig. 8. Moderate lemniscate (blue), circular (red), and tighter lemniscate (green) reference paths and their associated k_1 histories. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

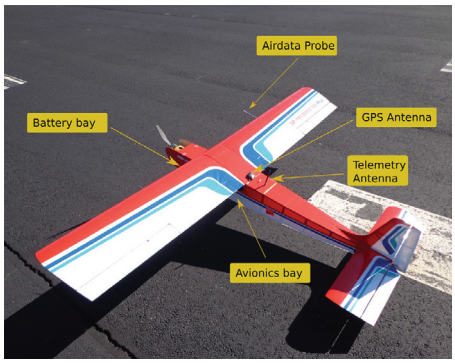


Fig. 9. UAS platform.

where $\boldsymbol{p}(\xi_i) = (N_{\text{ref}}(\xi_i), E_{\text{ref}}(\xi_i), H_{\text{ref}}(\xi_i))$ is expressed in $\mathcal{F}_I$. The minimum distance between any point $\boldsymbol{a}$ and the reference path R is then defined as

$$\text{dist}\,(\boldsymbol{a}, R) = \{\inf \|\boldsymbol{a} - \boldsymbol{b}\|_2 |\ \boldsymbol{b} \in R\}, \tag{23}$$

where $\|\cdot\|_2$ denotes the standard Euclidean norm. Given equations (22) and (23), the mean path error is defined as

$$\text{MPE} = \frac{1}{\bar{N}} \sum_{k=1}^{\bar{N}} \text{dist}(\boldsymbol{p}(k), R), \tag{24}$$

where $\boldsymbol{p}$ is the UAS location in $\mathcal{F}_I$, and $\bar{N}$ is the total number of measurements taken. The root-mean-square control effort is defined as

$$u_r = \sqrt{\frac{1}{\bar{N}} \sum_{k=1}^{\bar{N}} \bar{\boldsymbol{u}}(k)^T \bar{\boldsymbol{u}}(k)}. \tag{25}$$

The third metric used is the average path time (APT) to complete the given reference path.

6. UAS test platform

The UAS platform used in the flight tests is based on the Senior Telemaster Plus fixed-wing radio-controlled aircraft from Hobby Express (2017). The aircraft is electrically powered using lithium polymer batteries and has a conventional landing gear with a tailwheel. The sensor suite of the UAS consists of a barometric pressure sensor (MS5611) for altitude measurement, a differential pressure sensor (4525DO) for airspeed measurement, a satellite-based augmentation system (SBAS) enabled U-blox NEO-7 GPS module for position measurement, and a miniature MPU 6000 inertial measurement unit from Invensense. An in-house built five hole air-data probe is utilized for obtaining measurements of the angle of attack and the angle of sideslip, which are used in the development of the aerodynamic model. The barometric pressure sensor and the inertial measurement unit are housed inside the Pixhawk (described next), while the other sensors are connected to the Pixhawk through serial or I2C peripherals.

The Autopilot system consists of two components: a 3DR Pixhawk (Pixhawk, 2017) and a Gumstix Overo Fire (Gumstix, 2017). The Pixhawk uses open-source ArduPilot firmware which is customized for the UAS platform. The Gumstix runs a light-weight version of Linux called the Yocto project and provides sufficient computing power for running advanced control and visual data processing algorithms in near-real-time. The computational tasks in the autopilot are shared between the Pixhawk and the Gumstix. The Pixhawk is responsible for input/output tasks and redundancy management. While the Pixhawk is adequate for executing basic stabilization and navigation tasks, more computationally demanding control and navigation algorithms are executed in the Gumstix Overo COM. All the control algorithms presented in this work are coded using Python and executed on the Gumstix. The Gumstix receives sensor measurement data from and sends control commands to the Pixhawk through a serial connection. The UAS transmits data to the ground station through an XBEE 900 Pro, point-to-multipoint radio modem. The Ground Control Station (GCS) module uses the open-source APM Mission planner and displays flight parameters in real-time during flight tests. Fig. 9 shows the UAS platform with the important subsystems marked on it.

7. Results and discussion

7.1. Simulation results

Simulations are performed with the four different controllers presented in Section 4 under the conditions outlined in Section 5. In order for the simulation results to be statistically meaningful, each controller is tasked to track the reference path 1000 times consecutively. Simulation results are presented for the moderate lemniscate and the circular reference paths. The performances of the controllers are compared in terms of the three performance metrics described earlier, namely the mean path error, the control effort, and the average path time. Since the PID controller is only capable of tracking circular paths and straight line paths, the lemniscate path cannot be tracked as it has a continuously changing curvature. Therefore, to evaluate the performance of the PID controller, the lemniscate path is approximated as a polygon with 17 edges.

The distribution of the mean path error over 1000 circuits for the moderate lemniscate and circular paths is presented for each of the four controllers in Fig. 10. A cumulative distribution function (CDF)-like representation is used to succinctly display the position tracking performance of the controllers based on the simulation results. A summary of the values of the performance metrics is provided in Table 4.

All the four controllers are able to successfully complete the 1000 circuits. For both of the paths, the LPV controller has the lowest MPE. For the lemniscate path, the LPV controller has an MPE of 2.57 m with a standard deviation of $\sigma = 0.059$ m. While the LTI controller has a higher MPE compared to the LPV controller, it performs better than the RT and PID controllers. Although the MPE of the PID controller

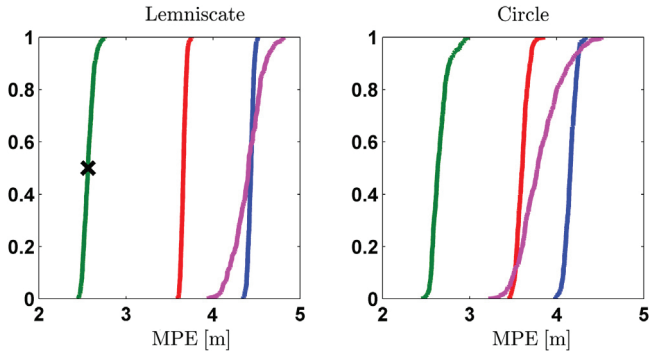


Fig. 10. RT, LTI, LPV, and PID MPEs over 1000 loops. The CDF-like representation is used to succinctly display the results. For example, the point marked by (x) on the plot has coordinates (2.564, 0.5), indicating that in 50% of the 1000 loops, the MPE value is less than or equal to 2.564 m. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Path-Following Performance Results Based on Simulation.

	Moderate lemniscate			Circle		
	MPE	u_r	APT	MPE	u_r	APT
RT	4.43 ± 0.033	0.243	153.44	4.16 ± 0.065	0.258	48.52
LTI	3.66 ± 0.030	0.249	153.33	3.61 ± 0.068	0.248	48.44
LPV	2.57 ± 0.059	0.279	142.20	2.65 ± 0.094	0.292	44.72
PID	4.31 ± 0.169	0.235	157.04	3.72 ± 0.253	0.236	49.08

Note: The values provided for MPE are the mean and standard deviation of the mean path errors over 1000 loops.

is comparable to that of the RT controller, the PID controller has a higher standard deviation of 0.169 m. The control effort exerted by the PID controller, however, is the lowest among all the controllers. The LTI and RT controllers have comparable values of u_r, which are lower than the control effort exerted by the LPV controller. It is found that in the presence of disturbances, the aileron and rudder input energies demanded by the LPV controller are higher in order to maintain better position tracking. Efforts to reduce u_r for the LPV controller, such as increasing the performance weights on the aileron and rudder control inputs and using different high-pass filters, have not been very successful and ended up increasing the MPE. It is inferred that the LPV controller's better position tracking capability comes at the expense of increased control effort. Among all the controllers, the PID controller has the highest APT of 157.04 s. The APT for the LTI controller is comparable to that for the RT controller, but the APT for the LPV controller is the lowest among the four. It is observed that the RT, LTI, and PID controllers have difficulty maintaining an appropriate airspeed and have a tendency to slow down. The LPV controller, however, is able to successfully maintain the desired airspeed in the midst of disturbances, and as a result yields a lower APT than the other controllers. A similar observation holds for the case of no disturbances. A possible reason for this behavior could be the use of parameter-varying trim points in the LPV control design, whereby the controller continuously corrects the trim states and control commands as the curvature of the reference path changes.

It can be seen from Table 4 that the qualitative observations made with respect to the lemniscate path also hold true for the circular path. The PID and RT controllers show the most improvement in MPE compared to their corresponding lemniscate mean path errors. The MPE values for the LTI and LPV controllers are similar to their lemniscate counterparts, with the LPV controller having a slightly higher MPE for the circular path than for the lemniscate path. While the u_r values for the LTI and PID controllers are similar to their corresponding values in the lemniscate case, the RT and LPV controllers have slightly higher values for u_r. As in the lemniscate case, the LPV controller maintains airspeed better than the other controllers and has the lowest APT, although the difference in APT is not as pronounced due to the shorter path length.

Table 5
Path-Following Performance Results from Flight Tests.

	Moderate lemniscate			
	MPE	u_r	APT	Wind (m/s)
RT	4.99	0.227	146.7	2.5 - 3.5
LTI	3.91	0.239	146.2	2.5 - 3.5
LPV	3.41	0.353	138.0	2.5 - 3.5
LTI	2.97	0.234	147.4	< 2.0
LPV	2.64	0.327	141.7	< 2.0
	Circle			
	MPE	u_r	APT	Wind (m/s)
RT	5.18	0.211	47.3	3.0 - 3.5
LTI	3.78	0.212	47.8	3.0 - 3.5
LPV	3.56	0.318	46.5	3.0 - 3.5
PID	4.71	0.171	46.5	3.0 - 3.5

For both paths, the circuit with the worst MPE is shown in Fig. 11 for each of the four controllers. The initial position of the vehicle is at $(0, 0)$, and the direction of the vehicle along the path is indicated in the figure by a black arrow. While bias in position tracking is evident for each controller due to the presence of wind, the RT controller additionally suffers from oscillations in the $\boldsymbol{x}_I \boldsymbol{y}_I$-plane. It is found that any attempt to damp out these oscillations for the worst-case circuit degrades the overall path-following performance of the RT controller.

For completeness, the performances of the controllers are studied in a flight regime different from the one used in designing the controllers. Specifically, simulations are performed at a different nominal flight condition, where the altitude and the airspeed are chosen as 660 m and 20 m/s, respectively. As expected, the mean path error increases for all four controllers due to the change in the operating flight condition. The increase in the MPE values, however, is smallest for the RT controller. A possible explanation could be the ability of the backstepping controller, which is used in the outer-loop of the RT controller, to adapt to the new flight condition more effectively than the other controllers. Moreover, it is noticed that the qualitative observations made in the previous paragraphs also hold true for this flight regime.

7.2. Flight test results

This section presents results from flight tests conducted using the UAS platform described in Section 6. The flight tests are conducted at the Kentland Experimental Aerial Systems (KEAS) laboratory located at Virginia Tech's agricultural research facility. To enable comparison between the controllers, multiple flights are performed with each controller to collect data at similar wind conditions. The wind speeds during the flight tests are obtained by subtracting the GPS derived ground speed from the airspeed measured by the airspeed sensor. A wind vane located near the airstrip provides the wind direction. The software implementation of each controller is first tested in a hardware-in-the-loop simulation (HILS) setup, which consists of the autopilot hardware interfaced with a laptop to simulate the UAS dynamics. The HILS verified controller code is then executed on the UAS.

A summary of the flight test results is provided in Table 5. Fig. 12 shows the position tracking performance of the controllers for the moderate lemniscate and circular paths under similar wind conditions. Flight test results of the PID controller for the lemniscate path are not provided as the UAS had large overshoots in position due to the polygonal approximation of the path. Hence, only flight test results for the circular path are provided for the PID controller. Similar to the results from the simulation studies, the LPV and RT controllers have the lowest and highest MPE, respectively, among all the controllers for both of the paths. However, the difference in MPE between the LPV and LTI controllers is not as significant as in simulation. For all the controllers, the MPE from flight tests is higher compared to simulation. This could be attributed to uncertainties in the inertial properties, throttle model

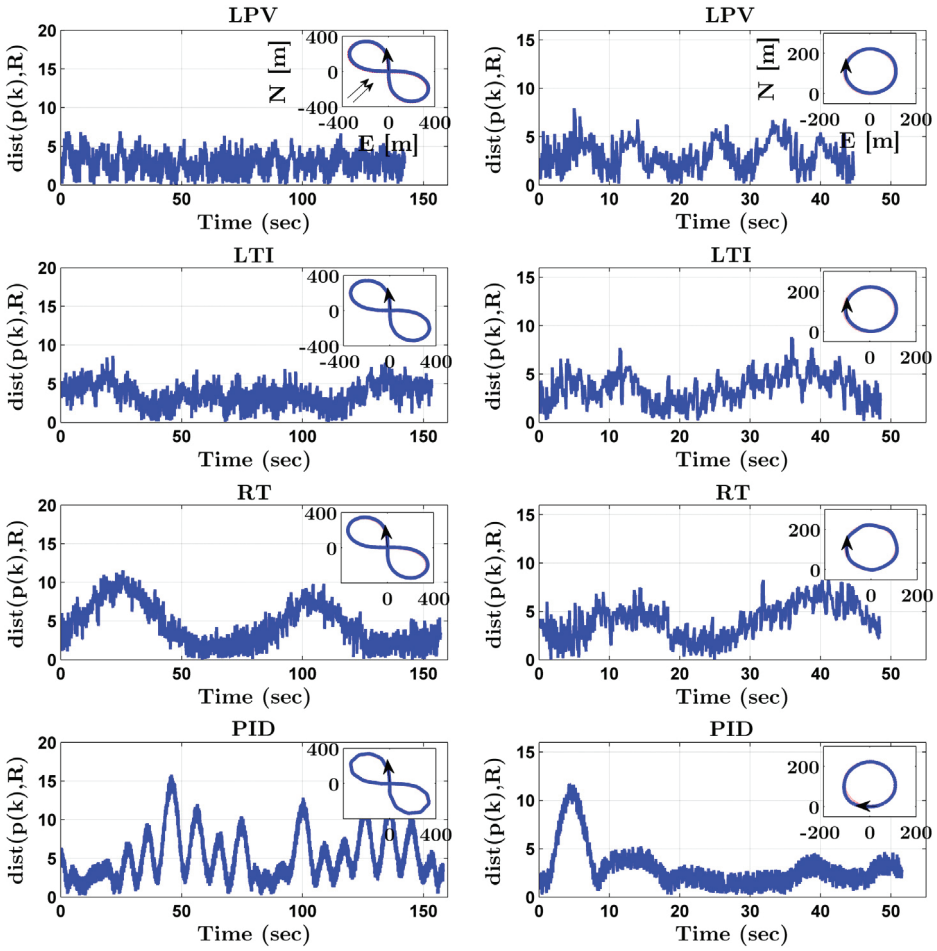


Fig. 11. The worst-case simulation runs for each controller in tracking the moderate lemniscate and circular paths, where $\mathrm{dist}(\boldsymbol{p}(k), R)$ is as defined in Eq. (23) and is expressed in meters.

and the aerodynamic nonlinearities, which are difficult to completely capture in the mathematical model of the UAS used in control design.

While the control efforts exerted by the RT and LTI controllers are comparable, the u_r for the LPV controller is higher in comparison by about 50%. The increase in u_r is due to the higher values of aileron and rudder control inputs demanded by the LPV controller. The PID controller is once again the most economical in terms of control effort. The APT to traverse the circular path is comparable for all the controllers, however, the LPV controller has lower APT for the lemniscate path in comparison to the LTI and RT controllers.

From the flight test results, it is observed that the performances of the LTI and LPV controllers are similar. This is not very surprising as the reference paths considered do not involve a sharp variation in k_1. The LPV controller is synthesized for arbitrary variations of k_1 subject to the condition $|k_1| \leq 0.0141$, while the LTI controller is synthesized by linearizing the lumped path-following and UAS dynamics about $k_1 = 0$. Thus, in order to fully explore the capabilities of the LPV controller, flight tests with the tighter lemniscate path shown in Fig. 8 are conducted, where the maximum value of k_1 is 0.0125. Fig. 13 shows the position tracking performance of the LTI and LPV controllers during flight tests for the tighter lemniscate path. In addition, the values of the performance metrics are summarized in Table 6. It is evident that the LTI controller struggles to execute the aggressive turn, resulting in rudder saturation which eventually leads to a controller failure. The LPV controller successfully completes the path without any failure in all the flight segments. Understandably, the MPE for the LPV controller in this case is higher than the corresponding values in the cases of the milder lemniscate and circular paths.

7.3. Discussion

It is apparent from the simulation and flight test results that no single controller can be adjudged as the best controller. Starting from the control design process, each controller has its own merits and limitations. The large number of tunable parameters involved is a distinct drawback of the RT and PID controllers. For example, in the case of the RT controller, there are 7 tunable constants that affect the resulting rate commands, some in non-intuitive ways. The addition of the $\mathcal{H}_\infty$ controller only compounds on this issue. The PID controller has 23 tunable parameters as shown in Table 3. By combining the UAS and path-following dynamics into one lumped system, the amount of tuning required during the control design process becomes more manageable. The application of a single $\mathcal{H}_\infty$ controller allows the control designer to more easily tune performance by directly relating inputs to the selected performance cost. However, it is found that the LPV controller is more sensitive to the problem formulation than the other three controllers. When the magnitudes of the disturbances are increased, for instance, the LPV controller must be re-tuned to achieve comparable performance levels, whereas the other controllers are found to be still adequately tuned. This is due in part to the fact that the controller design process is highly dependent on the control designer.

In terms of software implementation on the Gumstix microcontroller, the LTI and LPV controllers are by far the easiest to implement with minimum lines of code as they predominantly involve only matrix multiplications. Both the RT and PID controllers have an outer guidance loop and an inner stabilization loop, thereby making the controller code relatively lengthy. Moreover, the presence of inverse trigonometric

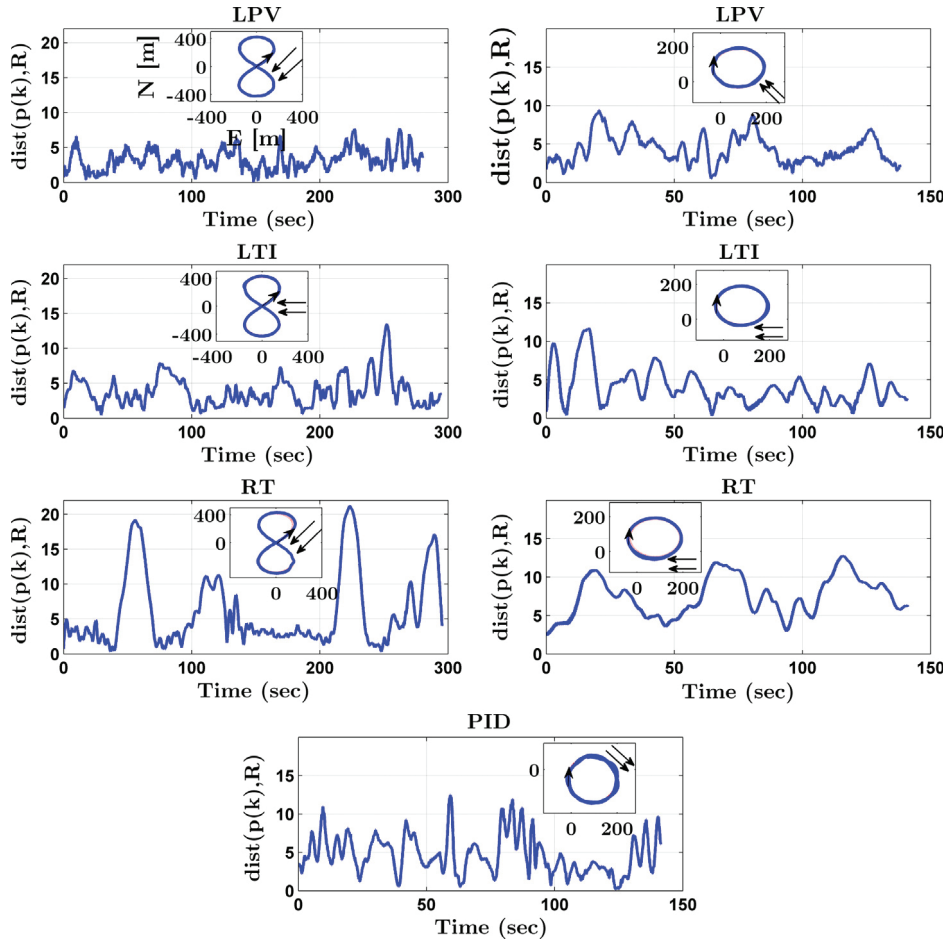


Fig. 12. Comparison of the different controllers from flight tests for the moderate lemniscate (2 loops) and circular (3 loops) paths; $\mathrm{dist}(p(k), R)$ is expressed in meters; the arrows represent the direction of wind; the wind speeds for the lemniscate flight tests are from 2.5 m/s to 3.5 m/s and the wind speeds for the circular flight tests are from 3.0 m/s to 3.5 m/s.

functions in the RT and PID controllers leads to an increase in the execution time.

The LPV controller has the most consistent position tracking performance and is capable of executing paths with sharp changes in curvature such as the tighter lemniscate. The improved position tracking performance achieved by the LPV controller can be attributed in part to the use of the reference path curvature as a gain scheduling parameter. This feature, along with the particular choice of the performance output in the ℓ_2-induced norm control setting, seems to yield an LPV controller that is capable of attenuating the effects of exogenous disturbances, such as steady winds, atmospheric turbulence, and sensor noise, on the position tracking errors much more effectively than the other controllers. However, the improved performance comes at the expense of higher control effort. For paths that do not involve sharp changes in curvature, the LTI controller has the best overall performance in terms of MPE, u_r, and APT. Although the RT and PID controllers require lower control effort, their position tracking performance is lower than that of the LTI and LPV controllers. Another drawback of the PID controller is that it cannot track paths with a continuous change in curvature such as the lemniscate.

Thus, the choice of a particular path-following control method for small fixed-wing UAS depends on such factors as performance specifications, the type of reference paths to be tracked, and the external environment in which the UAS is operating. It is believed that this work will provide some useful guidelines for the control designer to make such a choice.

Table 6
Path-Following Performance Results from Flight Tests for the Tighter Lemniscate.

	Tighter Lemniscate			
	MPE	u_r	APT	Wind (m/s)
LTI	7.98	0.295	93.1	< 2.0
LPV	5.16	0.342	87.8	< 2.0

Note: The values for the LTI controller are based on the flight segments where the controller did not fail.

8. Conclusions and future work

The use of a lumped path-following and UAS system and an $\mathcal{H}_\infty$ based robust control approach to the path-following problem was shown to yield improved performance compared to existing methods in the literature. While a direct comparison is difficult to make, the ease of implementation and the better path-following performance of the LTI and LPV controllers in comparison to the RT and PID controllers make them potentially valuable in application. Furthermore, the $\mathcal{H}_\infty$ framework makes it possible to apply various relevant robust control analysis results to the LTI and LPV controllers. The controllers designed within were shown to be sufficiently robust to sensor noise, disturbances, delays, and modeling inaccuracies, with over 200 hours of failure free simulated flight time, as well as several flight tests under different wind conditions. Additionally, the LPV controller exhibited good path-following performance for the tighter lemniscate with $k_{1_{\max}} = 0.0125$.

The results presented within are in no way intended to be general, and are highly dependent on the UAS being considered. They are even more dependent on the control designer, and other choices of

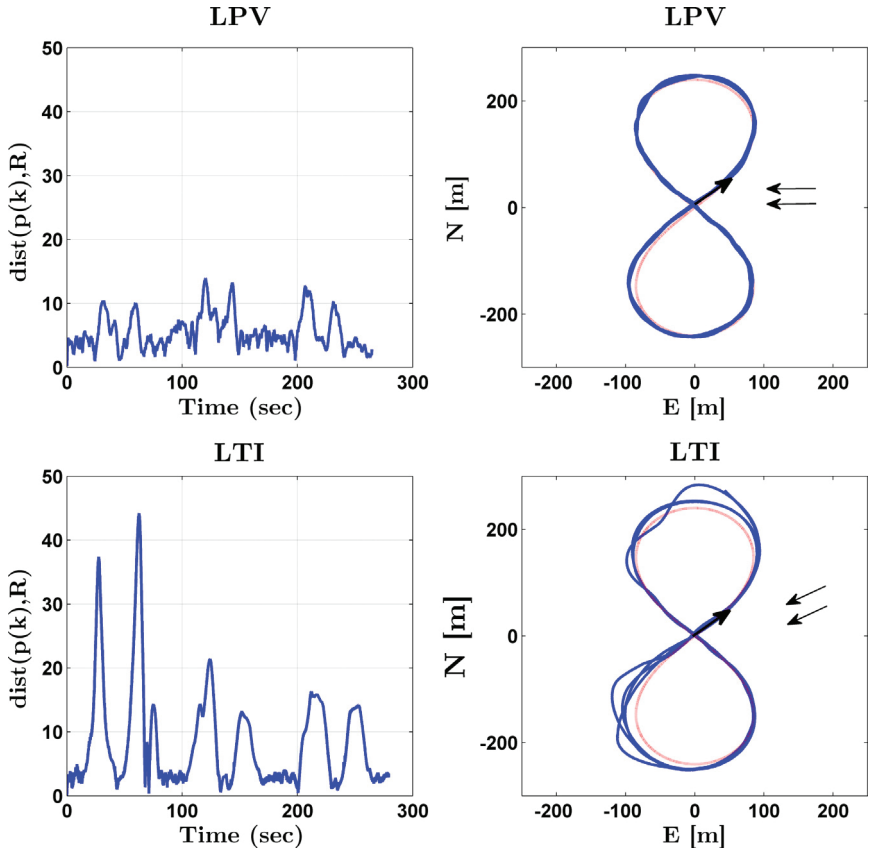


Fig. 13. Comparison of the LTI and LPV controllers from flight tests with the tighter lemniscate path (3 loops); $\mathrm{dist}(\boldsymbol{p}(k), R)$ is expressed in meters; the wind speeds during the tests are less than 2.0 m/s.

performance outputs or penalty weights may prove to produce better results for one or several of the control methods presented within. The intent of this work is to provide useful guidelines for designing path-following controllers for a small fixed-wing UAS.

Areas of future theoretical work include extending the combined path-following and UAS system to 3-dimensional paths, which would involve the inclusion of the $k_2(\ell)$ parameter in the path-following dynamics. Another area of future work is to use an airframe which is more agile than the Senior Telemaster to test paths with higher $k_{1_{\max}}$ than the ones considered in this work. Last, the authors plan to utilize a recently developed Integral Quadratic Constraint (IQC) analysis tool (Palframan & Farhood, 2016) to guide the control design process for the LPV controller with the goal of reducing the control effort typically demanded by such a controller.

Acknowledgments

This work was funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation sponsored industry/university cooperative research center (I/UCRC) under NSF Grant Nos. IIP-1539975 and CNS-1650465. The authors would also like to gratefully acknowledge the assistance from Micah Fry during the flight tests.

References

Abou Jaoude, D., & Farhood, M. (2017), Balanced truncation model reduction of nonstationary systems interconnected over arbitrary graphs, Submitted for publication to Automatica.

Aguiar, A. P., Hespanha, J. P., & Kokotović, P. V. (2007). Performance limitations in reference tracking and path following for nonlinear systems. *Automatica*, *44*(3), 598–610.

ArduPilot, Ardupilot Autopilot Suite. (2016a). http://ardupilot.org/ardupilot/.

ArduPilot, Ardupilot controller description. (2016b). http://ardupilot.org/plane/docs/common-tuning.html.

Arifianto, O., & Farhood, M. (2015a). Development and modeling of a low-cost unmanned aerial vehicle research platform. *Journal of Intelligent & Robotic Systems*, *80*(1), 139–164.

Arifianto, O., & Farhood, M. (2015b). Optimal control of a small fixed-wing UAV about concatenated trajectories. *Control Engineering Practice*, *40*, 113–132. http://dx.doi.org/10.1016/j.conengprac.2015.03.007.

Beck, C. L. (2006). Coprime factors reduction methods for linear parameter varying and uncertain systems. *Systems & Control Letters*, *55*(3), 199–213.

Beck, C. L., Doyle, J. C., & Glover, K. (1996). Model reduction of multidimensional and uncertain systems. *IEEE Transactions on Automatic Control*, *41*(10), 1466–1477.

Bishop, R. L. (1975). There is more than one way to frame a curve. *The American Mathematical Monthly*, *82*(3), 246–251.

Dačić, D. B. (2005). *Path-following: An alternative to reference tracking*. (pp. 1–118). University of California, Santa Barbara, Ph.D. thesis.

Doyle, J. C., Lenz, K., & Packard, A. (1986). Design examples using μ-synthesis: Space shuttle lateral axis FCS during reentry. In *Proceedings of the 25th Conference on Decision and Control* (pp. 2218–2223).

Farhood, M. (2012). Nonstationary LPV control for trajectory tracking: A double pendulum example. *International Journal of Control*, *85*(5), 545–562.

Gage, S. (2003). Creating a unified graphical wind turbulence model from multiple specifications. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*.

Gahinet, P., & Apkarian, P. (1994). A linear matrix inequality approach to $\mathcal{H}_\infty$ control. *International Journal of Robust and Nonlinear Control*, *4*(4), 421–448.

Gugercin, S., & Antoulas, A. C. (2004). A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, *77*(8), 748–766.

Gumstix, Gumstix Overo Fire. (2017), https://store.gumstix.com/coms/overo-coms/overo-firestorm-y-com.html.

Guthrie, K. T. (2013). *Linear parameter-varying path following control of a small fixed wing 58 unmanned aerial vehicle* (Master's thesis). Virginia Polytechnic Institute & State University.

Hanson, A. J., & Ma, H. (1995). *Parallel transport approach to curve framing*. (pp. 1–20). Bloomington, IN: Indiana University Department of Computer Science, Technical Report TR425.

Hepperle, M. Javaprop - design and analysis of propellers. (2006), http://www.mh-aerotools.de/airfoils/javaprop.htm.

Hobby Express, Senior Telemaster Plus. (2017), http://www.hobbyexpress.com/senior_telemaster_plus_1034837_prd1.htm.

Kaminer, I., Pascoal, A., Xargay, E., Hovakimyan, N., Cao, C., & Dobrokhodov, V. (2010). Path following for small unmanned aerial vehicles using L1 adaptive augmentation of commercial autopilots. *Journal of Guidance, Control, and Dynamics*, *33*(2), 550–564.

Kaminer, I., Yakimenko, O., Dobrokhodov, V., Pascoal, A., Hovakimyan, N., Cao, C., Young, A., & Patel, V. (2007). Coordinated path following for time-critical missions of multiple UAVs via L1 adaptive output feedback. In *AIAA Guidance, Navigation, and Control Conference and Exhibit* (pp. 1–34).

Löfberg, J. (2004). YALMIP: A Toolbox for Modeling and Optimization in MATLAB. In *Proceedings of the CACSD Conference*.

Moorhouse, D. J., & Woodcock, R. J. (1982). *Background information and user guide for MIL-F-8785C, military specification - flying qualities of piloted airplanes*. (pp. 1–244). Wright-Patterson Air Force Base, Ohio: Air Force Wright Aeronautical Laboratories, Tech. rep..

Nelson, D. R., Barber, D. B., McLain, T. W., & Beard, R. W. (2006). Vector field path following for small unmanned air vehicles. In *Proceedings of the 2006 American Control Conference* (pp. 5788–5794).

Osborne, J., & Rysdyk, R. T. (2005). Waypoint guidance for small UAVs in wind. In *AIAA Infotech@Aerospace 2005 Conference* (pp. 1–12).

Packard, A. (1994). Gain scheduling via linear fractional transformations. *Systems & Control Letters*, *22*, 79–92.

Palframan, M. C., & Farhood, M. (2016). An IQC analysis framework for small fixed-wing UAS. In *American Control Conference*.

Palframan, M. C., Guthrie, K. T., & Farhood, M. (2015). An LPV Path-Following Controller for Small Fixed-Wing UAS. In *Proceedings of the 54th Conference on Decision and Control* (pp. 1–6).

Palframan, M. C., & Woolsey, C. A. (2014). UAS source localization with high latency sensors in turbulent environments. In *AIAA Guidance, Navigation, and Control Conference* (pp. 1–15).

Park, S., Deyst, J., & How, J. P. (2004). A new nonlinear guidance logic for trajectory tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit* (pp. 16–19). Providence, Rhode Island: AIAA.

Pixhawk, Pixhawk Autopilot hardware. (2017), https://pixhawk.org/modules/pixhawk.

Raol, J. R., & Singh, J. (2009). *Flight Mechanics Modeling and Analysis*. ISBN: 97814200 67538, Boca Raton: CNC Press.

Rysdyk, R. T. (2006). UAV path following for target observation in wind. *Journal of Guidance, Control, and Dynamics*, *29*(5), 1092–1100.

Soetanto, D., Lapierre, L., & Pascoal, A. (2003). Adaptive, non-singular path-following control of dynamic wheeled robots. In *Proceedings of the 42nd IEEE Conference on Decision and Control*. ISBN: 0780379241, (pp. 1765–1770).

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, *11*(1–4), 625–663.

Sujit, P., Saripalli, S., & Sousa, J. (2014). Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems*, *34*(1), 42–59.

Wise, R. A., & Rysdyk, R. T. (2006). UAV coordination for autonomous target tracking. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit* (pp. 1–22).

Yang, J.-M., & Kim, J.-H. (1999). Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, *15*(3), 578–587.