

SIMULATION-BASED AUTOPILOT TUNING FOR FIXED-WING UAVS

Cezar-Victor Borugă[1], Natalia Zalewska[1], David Lundström[1] & Petter Krus[1]

[1]Linköping University, Department of Management and Engineering, Fluid and Mechatronic Systems
SE-581 83 Linköping, Sweden

Abstract

In-flight tuning of control systems carries risks, making simulation-based methods an attractive alternative for large-scale Unmanned Aerial Vehicles (UAVs). This study proposes and evaluates a process built around the X-Plane 12 flight simulator for initial control loop tuning of fixed-wing UAVs equipped with ArduPilot. Subscale flight testing is used to evaluate the accuracy of an aircraft model built within the simulator and to compare autopilot performance between simulation-based and in-flight tuning. The results show that the simulation produced usable controller gains, comparable to those obtained through in-flight tuning, despite some inaccuracies in the flight model. The same tuning process is then successfully demonstrated on a large-scale UAV. These findings indicate that X-Plane 12 could serve as an expedient, low-cost tool for preliminary autopilot tuning of fixed-wing UAVs, reducing the risks and costs of large UAV development.

Keywords: UAV, Tuning, Simulation, Flight Testing

1. Introduction

The development and testing of large unmanned aerial vehicles (UAVs) present significant challenges in terms of cost and safety. Some of these challenges arise from the inherent risks associated with initial flight testing of prototypes, where control systems are often not fully tuned.

Conventionally, UAV control systems are tuned through in-flight adjustments during early test flights [1] or through simulations using system identification models based on in-flight data [2]. While these approaches are standard for small platforms like the one shown in Figure 1a, they are often impractical for larger aircraft such as the one in Figure 1b. In-flight tuning of large UAVs carries significant safety risks, whereas system identification campaigns are time-consuming and costly. In addition, both methods consume valuable flight hours.



(a) Skyhunter 1800, a 1.8 meter wingspan UAV



(b) F720, a 7.2 meter wingspan UAV

Figure 1 – UAV platforms considered in this study.

In such cases, predictive modeling and simulation stand out as a promising avenue, offering a sandbox for tuning and testing devoid of real-world risks. Unlike the in-flight tuning and system identification methods, which rely on actual flight data and real-world testing, predictive modeling uses theoretical principles and statistical data to simulate aircraft behavior.

The literature on fixed-wing aircraft is rich with predictive flight dynamics models. The main source of variation among these models lies in the chosen method for predicting aerodynamic derivatives. Options range from the semi-empirical geometric build-up techniques, such as those provided by Raymer [3] and Roskam [4], to numerical approaches. Numerical tools that utilize Vortex Lattice or Panel Methods like OpenVSP, XFLR5, and Athena VLM are popular for rapid estimations. Navier-Stokes solvers are also used to provide higher fidelity results in exchange for higher computational costs [5], [6]. Alternatively, a method often employed in the UAV-centric literature is the use of commercial flight simulators such as X-Plane and Microsoft Flight Simulator.

Comparisons between different predictive modelling methodologies are frequently presented in literature. However, studies assessing the accuracy of the resulting models against real flight testing data are rare, especially for UAV-related applications. The few exceptions that do feature empirical validation, such as the works of Leong [2] and Johnson [5], offer only a brief glance at how such models can be leveraged for control loop tuning. As a result of this literature gap, there is limited guidance on which predictive modelling methodology is most appropriate for this application, and it remains unclear how much confidence can be placed in control loop tuning results obtained using predictive modelling approaches.

This issue was encountered during a project involving the F720 UAV shown in Figure 1b. The F720 has a maximum take-off mass of up to 250 kg and is subject to considerably more stringent operational regulations than small UAV platforms. After a flight control systems retrofit, initial in-flight controller tuning was not a viable option due to risks of damage or loss of control as well as operational constraints. This created the need for an initial controller tuning approach capable of providing, at minimum, a rough but functional set of controller gains prior to flight testing.

To address this need, a predictive simulation-based approach was built around X-Plane 12. This choice was primarily driven by modelling expediency considerations. X-Plane 12 provides aerodynamic, propulsion, inertial, and atmospheric models, reducing aircraft modelling effort compared to alternative approaches. In addition, the built-in support for real-time interfacing with ArduPilot's Software-In-The-Loop (SITL) firmware allows direct integration with the autopilot used in this study.

This paper investigates the use of X-Plane 12 as a low-cost tool for simulation-based initial tuning of fixed-wing UAV control systems. A modeling and tuning process centered around the simulator is proposed and used to establish baseline controller parameters. This process is first evaluated and validated using a subscale aircraft by comparing the performance of simulation-based tuning with that of in-flight tuning. Following this validation, the same framework is demonstrated on a full-scale UAV.

2. Theory

Both aircraft presented in this study utilize ArduPlane, ArduPilot's firmware for fixed-wing aircraft. Arduplane's control system consists of multiple controllers, each dedicated to specific functionalities. Arguably the most important control loops in a fixed-wing UAV are the attitude control loops [1], [7]. These loops control the orientation of the aircraft. In ArduPlane, all higher-level controllers (airspeed, altitude, turn) rely on the attitude control loops to function.

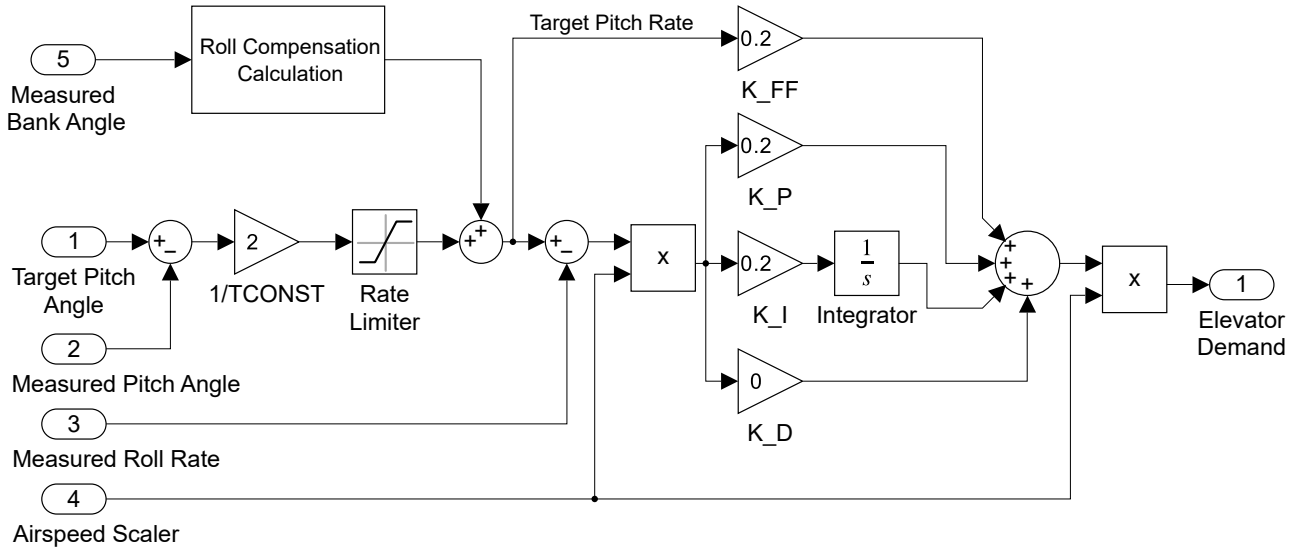


Figure 2 – Arduplane's Pitch Angle Controller, based on [1]

Arduplane's Pitch control loop is depicted in Figure 2. Despite being commonly referred to as the *"Pitch Attitude Loop"* or *"Pitch Angle Loop"*, the PID (Proportional, Integral, Derivative) and FF (Feed-Forward) terms utilize angular velocities as inputs rather than angles. If we disregard all the filters and limiters and consider the *1/TCONST* term as just another gain proportional to the angle error, we can think of this controller as a cascaded P and PIDFF loop. Here, the PIDFF loop controls the angular velocity, while the P loop controls the attitude. The Roll angle controller architecture is identical to the Pitch angle controller, except for the *Roll Compensation Calculation* block, which is not present.

The yaw controller can be set up in two different ways, depending on the type of aircraft. The first mode acts as a combined yaw damper and sideslip controller for standard, non-aerobatic planes. The second mode is a pure yaw rate controller designed for aerobatic aircraft, operating similarly to the pitch and roll architectures mentioned above. For the purposes of this paper, the yaw rate controller will not be used. The parameters for tuning the coordinated turn and sideslip controllers are set manually based on in-flight behavior, as described in Ardupilot's documentation [1].

For a fixed wing aircraft, the pitch attitude controller has the highest potential to cause damage if mistuned, as overly aggressive elevator commands can cause structural damage. By comparison, the roll and yaw controllers pose less potential for overstressing the airframe, but can still cause loss of control if not correctly tuned.

In this paper, we will make extensive use of Arduplane's Fly-By-Wire-A, or FBWA, flight mode. This flight mode enables the Angle Control loops presented in this section. Hence, this is sometimes referred to as *Angle Mode*. In this mode, the operator's joysticks set the airplane's target pitch and roll angles. This means that the UAV operator has no direct control over the ailerons and elevator, and instead the autopilot will use the angle control loops to command control surface deflections to achieve the set target.

3. Methodology

This section details the proposed simulation framework, aircraft models, and controller tuning strategy. Additionally, it establishes the methodology for validating the process using a low-risk subscale aircraft and then scaling it up to a full-scale platform.

3.1 Simulation Setup

Once the simulation environment has been chosen, it must be coupled to the autopilot. In real world deployments, ArduPilot typically runs on Flight Controllers with ARM-based processors, sensor suites, and various Input/Output (I/O) interfaces. While modeling every subsystem on board a UAV or using a Hardware-in-the-Loop (HITL) simulation architecture is technically possible, it would be unnecessarily time-consuming for the purposes of this work. ArduPilot's Software-in-the-Loop (SITL) firmware is almost identical to the Flight Controller native firmware, with the primary differences concerning the Extended Kalman Filter and I/O interfaces [5].

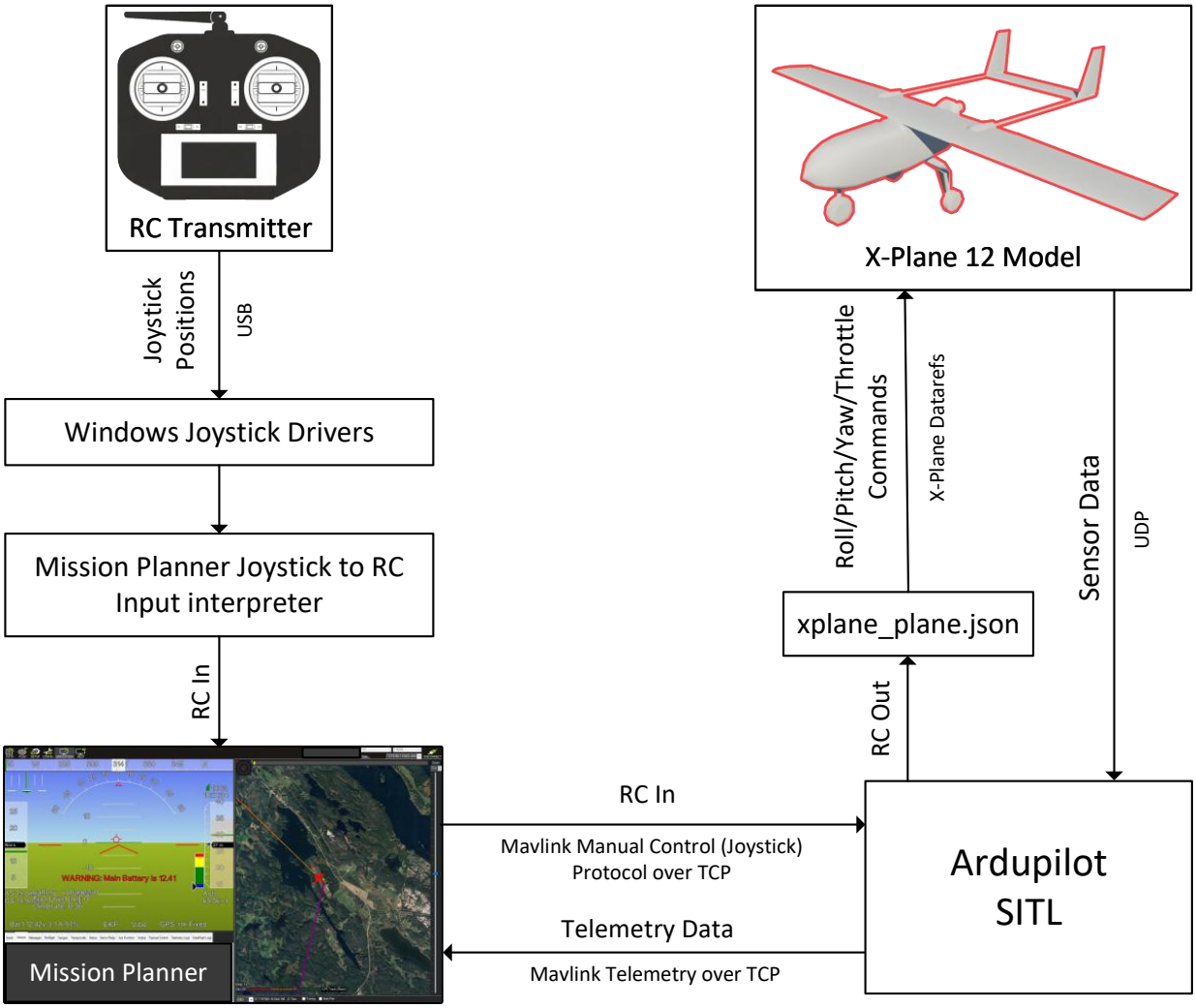


Figure 3 – Simulation Framework

Figure 3 depicts the simulation framework used for both subscale and large scale models. Everything except the RC Transmitter executes on the same computer running Microsoft's Windows 11 operating system. The file *xplane_plane.json* is a translator between Ardupilot's typical actuator command signals and X-Plane's pilot inputs. At the time of writing, this is the simplest setup we could devise that still allows manual pilot input. The downside of sending pilot inputs via Mission Planner and Mavlink is the addition of around $200\,ms$ of input lag. This is not the recommended solution by ArduPilot, but a workaround to bypass some issues with X-Plane 12's input interpreter. While this delay does not affect the results of this paper, it would negatively impact tasks such as simulator-based pilot training.

3.2 Tuning of the Attitude Control Loops

Arduplane's documentation [1] is extensive regarding the tuning of attitude control loops. In essence, a successful tune is achieved after setting all airplane-specific parameters to correct, realistic values and then adjusting the gains for the rate loops, in this order.

The airplane-specific parameters primarily relate to angular velocity limits, time constants, and turn coordination adjustments. Due to these parameters affecting, among other things, the aggressiveness of the maneouvers commanded by the autopilot, it is important to set them correctly to avoid over-stressing the airframe. Normally, these limits are determined by analyzing a series of maneouvers performed under manual pilot control, then uploaded to the autopilot.

The angular velocity loops can be tuned either manually or by using Arduplane's autotuning algorithm. For the purposes of this paper, we will rely on the autotuning algorithm to produce a usable initial tuning. Manual fine tuning is beyond the scope of this work.

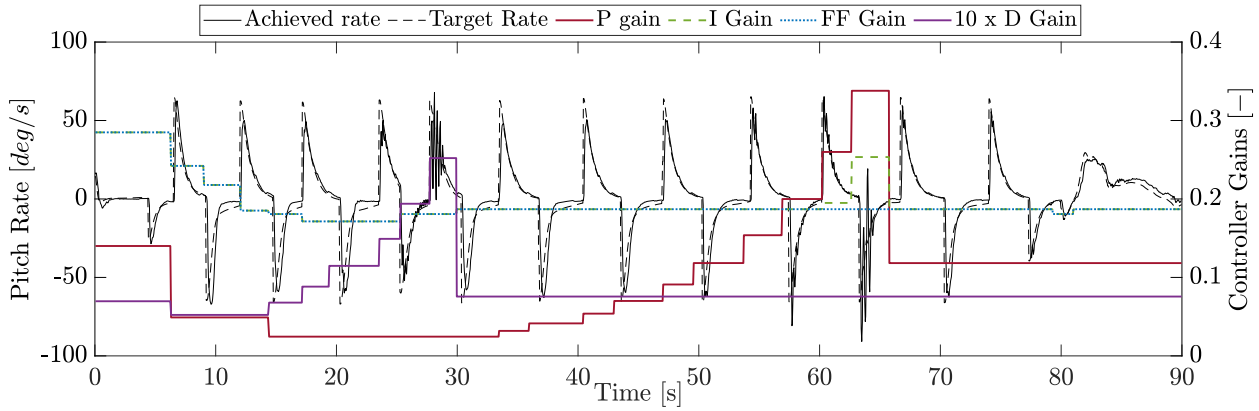
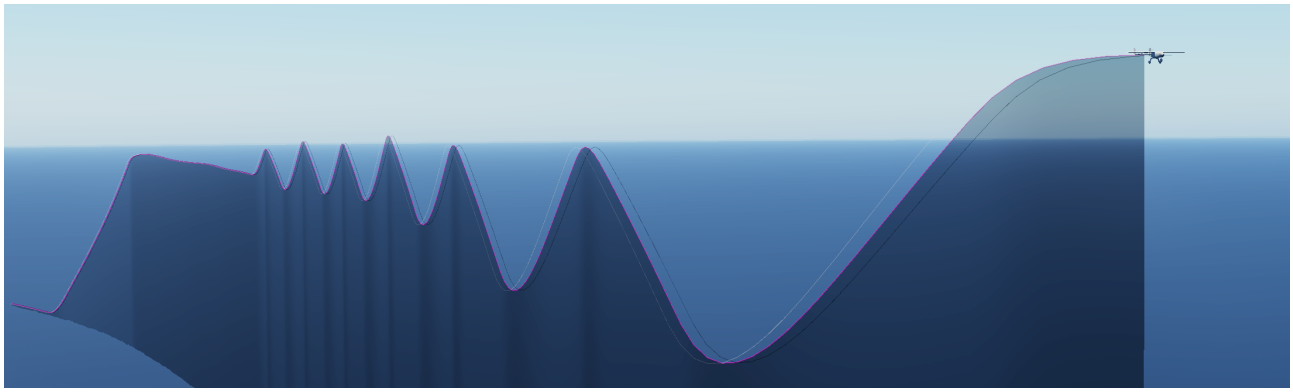


Figure 4 – Evolution of the pitch rate Controller gains during an autotuning flight

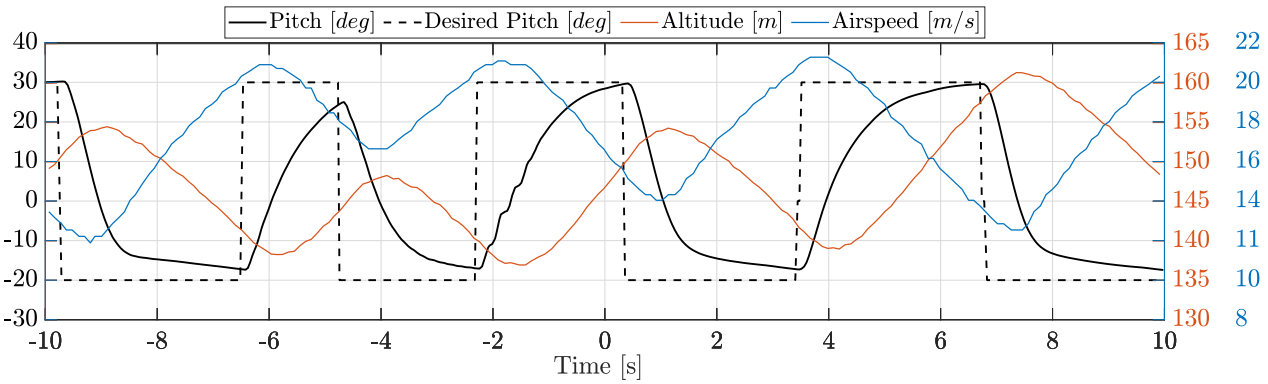
Figure 4 depicts the evolution of the pitch rate controller gains during a typical pitch autotuning procedure. When the autotuning procedure begins, the default gains (unless changed) are used in the controller. During this procedure, sharp maneuvers are performed and the response of the aircraft is observed by the autopilot. First, the FF gain is set, then the Derivative gain and lastly the Proportional gain. The Derivative term is set to a third of the value that produced oscillations. The Integrator gain is set to be equal to the FF gain unless the Proportional is much higher than the FF. The autotuning algorithm for the Roll axis is similar to that of the Pitch axis.

An important aspect of autotuning is its indifference to maneuver direction, as the algorithm only observes absolute achieved and target rates. This is a non-issue for roll autotuning because airspeed remains relatively constant. However, because airspeed varies when pitching up versus down, pitch autotuning is highly sensitive to how the plane is being flown during the process. This variation can cause gains to be set at different airspeeds. For example, in Figure 4, the P gain was set after a pitch-down while the D gain was set during a pitch-up. To reduce the effect of this, we experimented with several different flight profiles for pitch autotuning. The one that produced the smallest spread in results is depicted in Figures 5a and 5b. Pitch-up and pitch-down maneuvers are performed back-to-back, keeping all maneuvers within a narrow airspeed range close to cruise speed. Roll axis tuning was also executed with back-to-back maneuvers, but solely to reduce flight time.

Another factor affecting autotuning results is whether the roll and pitch maneuvers overlap. Performing simultaneous multi-axis maneuvers increases the spread of the resulting gains, whereas a sequential approach produces more consistent results. Therefore, during autotuning runs, roll and pitch maneouvers were not overlapped. Note, however, that the maneouvers can be interwoven (3-4 pitch maneouvers interrupted by 3-4 roll maneouvers, for example), as the autotuning algorithm only takes into account large inputs and ignores small corrections.



(a) Trajectory of the aircraft during pitch autotuning



(b) Pitch angle, altitude and airspeed during pitch autotuning

Figure 5 – Example of a pitch autotuning procedure

Despite all of these precautions, the autotuning process never produces two identical sets of gains, even in a clean simulation environment. This is due to minute variations in airspeed, timing of pilot input, sensor noise and other factors. One can think of the autotuning process as a series of measurements subject to noise. It is entirely possible that performing a single autotuning run will produce controller gains that are safe to operate, but an outliner - either more aggresive or tamer than intended. Most of the time, for small UAVs, this is acceptable.

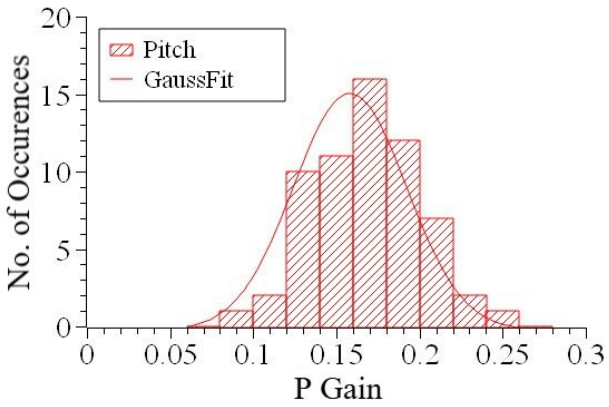


Figure 6 – Typical distribution of Pitch P gains after a series of autotuning runs

For higher risk applications, however, we deemed the possibility of using a set of "outlier" gains not acceptable. In order to provide a solid foundation for comparisons and more confidence in the tuning results, we chose to perform multiple autotuning runs for each aircraft. In principle, the distribution pattern of the data should align with a Gaussian distribution, as depicted in Figure 6. The peak of the Gaussian distribution can be interpreted as the intended result of the autotuning algorithm, and as such will be considered representative. It is worth noting that performing a large amount of autotuning runs is only time-efficient in a simulation environment, where the process can be largely automated. Gathering a statistically significant sample of real, in-flight autotuning runs, on the other hand, is often not practical due to time constraints.

3.3 Small-Scale Validation Strategy

The goal of this paper is not to build a high-fidelity model or to exactly replicate gains originating from real-world tuning. The objective is to produce a usable controller tune that reduces operational risk. In this context, validation means demonstrating that the proposed simulation and tuning process yields gains that have a high likelihood of being safe on a real aircraft. For our application, achieving this with a small-scale UAV provides the necessary confidence to attempt using the method on a larger scale platform.

Despite a high-accuracy model not being a direct requirement, the model should still behave reasonably close to the real aircraft. To verify this, we applied step aileron and elevator deflection inputs in both simulation and real flight, executing maneuvers similar to the examples in Figure 7. Before the maneuver is performed, the airplane is allowed to settle in straight and level flight in Manual mode. The step inputs are triggered by the pilot via a switch on the RC transmitter.

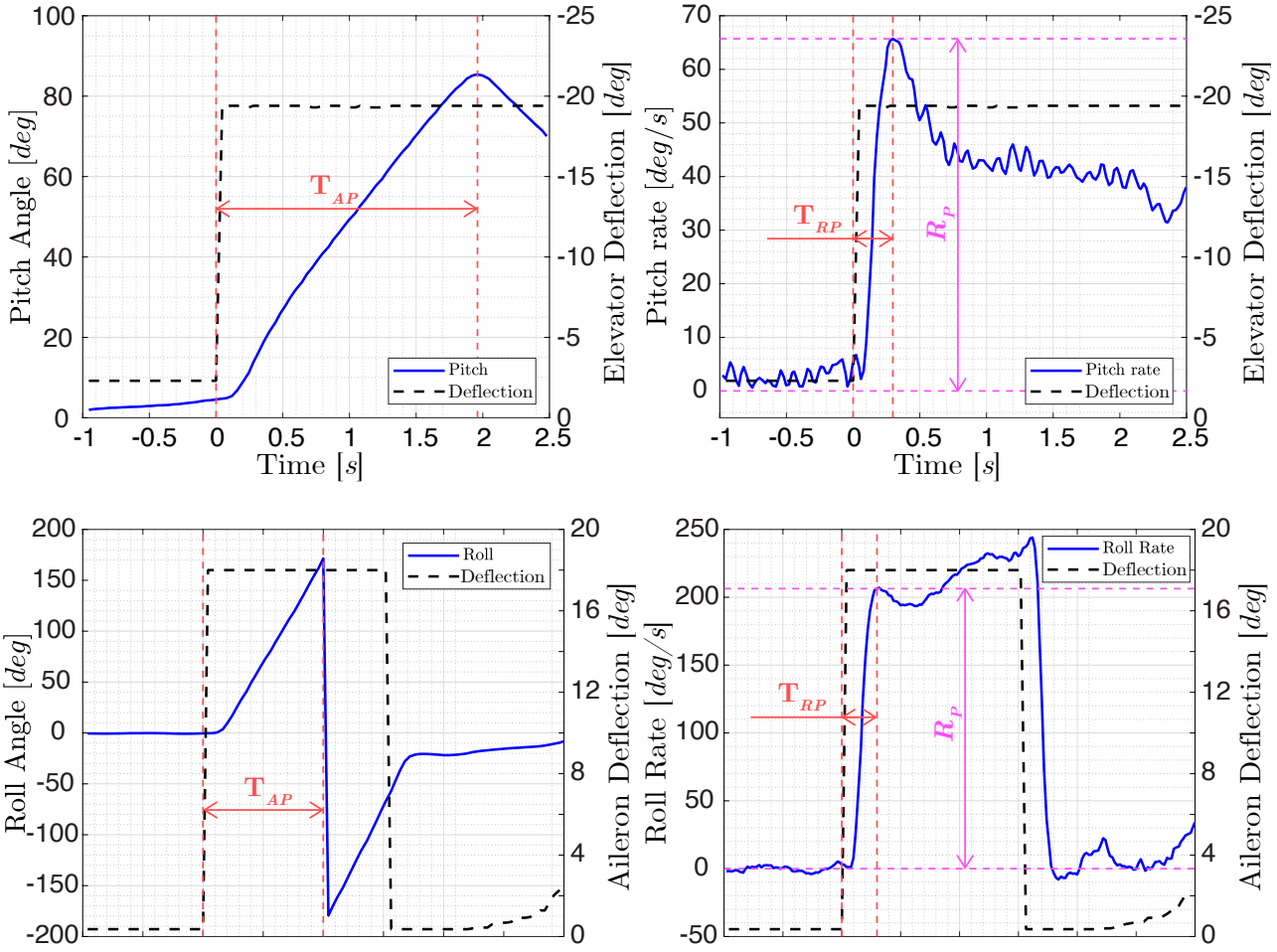


Figure 7 – Typical responses to step commands of elevator/aileron step. Top: Pitch, Bottom: Roll

Such aggressive commands are atypical of regular operation, but they are useful for qualitative comparisons and for determining the maximum and sustained pitch and roll angular velocities. To quantify the resulting responses, the following parameters are measured: Time to Angle Peak (T_{AP}), defined as the time to reach a vertical orientation ($90°$ pitch) or an inverted position ($180°$ roll) from the start of the input signal; Time to Rate Peak (T_{RP}), representing the duration from the start of the input signal to the first angular velocity peak; and Rate Peak (RP), which is the maximum angular velocity of the initial rate peak. These metrics are depicted in Figure 7. It is important to note that because the UAV lacks control surface position sensors, the deflection values plotted are based on commanded servo

inputs rather than measured control surface positions.

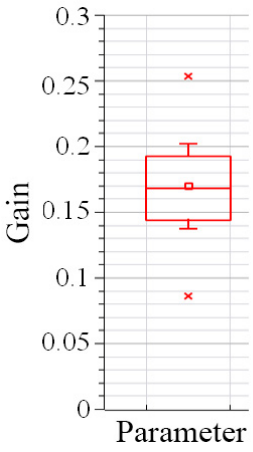


Figure 8 – Box Plot Example

The controller gains resulting from simulation-based tuning are expected to differ from those obtained during real flights. To determine how the differences in controller gains affect the performance of the control system of a real aircraft, a single representative gain set is selected from each environment and tested. To analyze the spread and select these final values, the tuning data is compiled into box plots, such as the one in Figure 8. The central box represents the interquartile range, where a horizontal line indicates the median and a central square marks the mean. The whiskers extend to one standard deviation above and below the mean, while the outer crosses denote the minimum and maximum bounds.

Once representative controller gain sets are selected, the most direct way of measuring the impact of simulation-based tuning is through a performance comparison. For this, we will measure the responses of the autopilot's angle controllers when using simulator-obtained controller gains compared to when using in-flight obtained gains. This will be done by evaluating the step responses of the aircraft to pitch and roll angle demands while operating in Ardupilot's FBWA mode. The step-response characteristics are measured using the maneuvers depicted in Figure 9. The pitch and roll angle commands are sent while the aircraft is flying straight-and-level at constant airspeed.

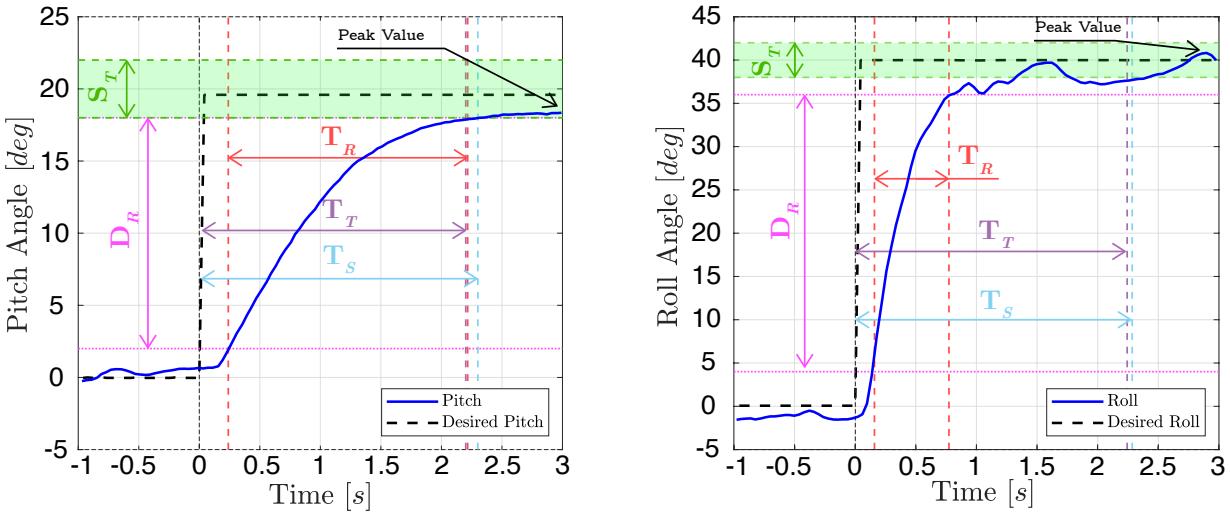


Figure 9

The rise time (T_R) represents the time required for the response to climb the rise distance (D_R), defined from 10% to 90% of the target change . The Angle Peak (AP) is the maximum value achieved by the response within the measurement window. Time to Angle Peak, (T_{AP}), is the time to reach the Angle Peak. The percentage overshoot (OS) or undershoot (US) is measured in relation to the commanded signal. Settling time (T_S) and transient time (T_T) determine when the aircraft enters a specific error tolerance band (S_T). This settling threshold is defined as $\pm 5\%$ of the commanded value for the roll axis and $\pm 10\%$ for the pitch axis. However, because pitch angle steps force the aircraft to trade airspeed for altitude under constant throttle, a steady state cannot be maintained. As such, transient and settling time are not relevant to measure for pitch maneouvers. These performance metrics are measured for multiple maneuvers. The numerical results will be presented as the averages of those maneouvers starting within $\pm 1 m/s$ of the intended starting airspeed.

3.4 Subscale Simulation Model

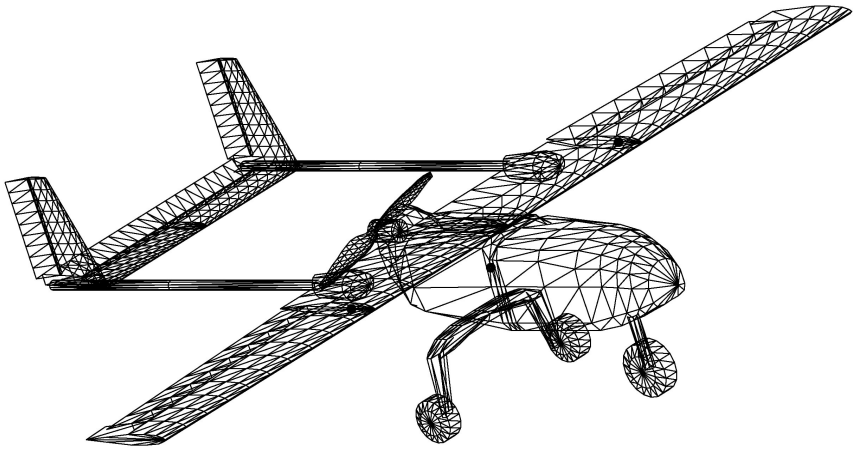


Figure 10 – Skyhunter 1800 X-Plane 12 Model

The model of the Skyhunter 1800 pictured in Figure 10 was created using Plane Maker, a utility for creating aircraft models for X-Plane 12. When using this utility, the modeling process is centered around providing data for the aerodynamic, propulsion, and inertial models baked into X-Plane. With a few exceptions, the modeling process followed the instructions from Plane Maker's manual [8].

The geometrical data required was mostly obtained by measuring the real aircraft. The only exceptions was the propeller geometry, which was obtained from the propeller manufacturer. The geometry definition followed Plane Maker's manual, with one notable exception: the outboard sections of the wing. One oddity of this particular airplane is the asymmetry of the aileron deflections. Due to poor construction, not only do both deflect more in one direction than the other, they also have slightly different maximum/minimum deflections. Since Plane maker does not allow this with the *"wing"* type elements, the outboard sections of the wing are of *"misc. wing"* type elements with two separate aileron instances. Based on our testing, this affects nothing except some of the diagnostic data displayed at runtime in X-Plane.

The geometrical data for the model's airfoils was obtained from public libraries [9] and modified to match the trailing edge thickness of the airplane's surfaces. Airfoil performance data was estimated by running X-Foil analyses within XFLR5 and then manually imported into Airfoil Maker, X-Plane's airfoil data utility.

The control surface minimum/maximum deflections were measured using a digital level. The average slew rates were estimated by applying step inputs to the actuators and extracting the total travel time from high-speed video recordings while the aircraft was stationary. The Ardupilot SITL firmware was configured to match the real setup as closely as possible. This is particularly important when it comes to the endpoints of the command signals sent to the servomotors powering the control surfaces. The Pulse Width Modulation (PWM) signal endpoints were matched to those of the real aircraft. Based on our experimentation, both the control surface deflection limits and PWM signal endpoints need to match to obtain representative tuning results.

The inertial properties of the airplane were estimated using a simplified CAD model. Prior to assembly, each individual part of the aircraft was weighed. The mass of adhesives was distributed based on the glued surface area. The Center of Gravity (CG) location was determined using a CG scale, a tool common to aeromodeling. The airplane was assumed to be laterally symmetrical.

The propulsion system performance is based on manufacturer specifications. The drag of the aircraft was based solely on prior observations of in-flight performance and engineering assumptions.

3.5 Large Scale Application

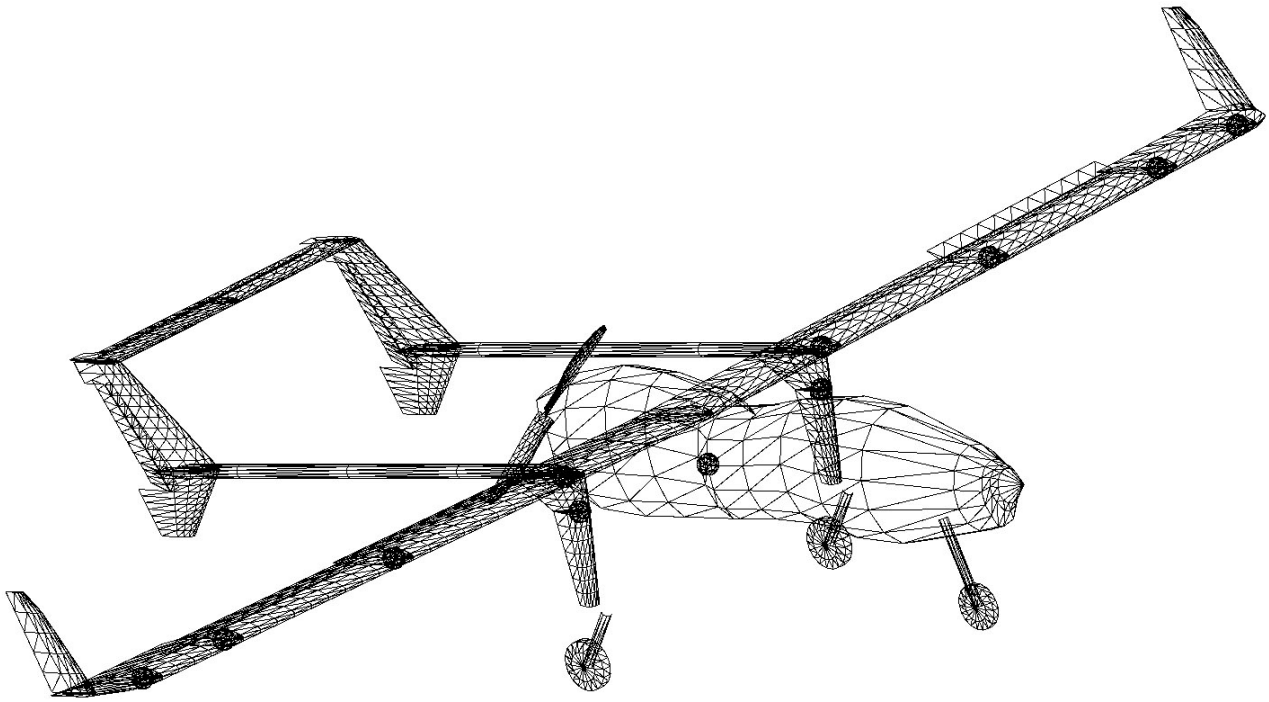


Figure 11 – F720 X-Plane 12 Model

After the subscale validation of the proposed modeling, simulation and tuning process, almost the same workflow was applied to the larger-scale F720, resulting in the model from Figure 11. While most of the process remained unchanged, refinements were made to address scale-specific issues that could be safely overlooked during subscale testing, while also incorporating lessons learned from the subscale study.

Subscale aircraft can typically survive much higher load factors than full-scale vehicles, so structural limits were almost completely ignored during subscale flight testing. On the other hand, the F720 tuning process included explicit assessments to ensure the control system would not overload the airframe during aggressive maneuvers at near-stall, cruise, and maximum airspeeds. Additionally, the real F720 is powered by a two-stroke gasoline engine that generates significant vibrations. Because controller derivative terms are highly sensitive to high-frequency noise, the derivative (D) gains for the roll and pitch rate controllers were intentionally set to zero for initial flights, overriding the non-zero values produced by the simulation-based tuning proocess.

Lessons from the subscale validation study also led to a change in how we identify vehicle performance limits. To determine the maximum achievable pitch rates, step elevator commands were abandoned because they can produce misleading results and high-speed stalls. Instead, pitch-up maneuvers were conducted using varying elevator deflection rates. By testing gradually increasing elevator command rates, the true maximum pitch rate before stall onset was identified, and with it the appropriate elevator application rate. A safety margin was also applied to reduce the maximum demanded roll and pitch rates, in case the simulator overestimated the maximum achievable rates.

A peculiarity of the X-Plane 12 version (12.0.9-rc-5) used for this paper is the estimation of downwash over the horizontal tail. This was noted after an excessive amount of elevator deflection was required to trim the initial simulator model of the F720. After a brief inverstigation, we found that X-Plane 12 overestimated the downwash angle by almost 2° compared to XFLR5, and between 1° and 1.5° compared to handbook methods [10], [11], [12]. To compensate for this, the mouting angle of the entire horizontal stabilizer was adjusted by 1.5°.

4. Results

4.1 Subscale UAV

4.1.1 Responses to manual control inputs

The pitch response of the subscale aircraft shows good similarity between simulation and real flight, as seen in Figure 12. The higher peak rate in the simulator occurs mostly because it assumes a perfectly rigid tail. In reality, the aircraft's flexible tail structure reduces elevator effectiveness due to aeroelastic effects. In addition to this, differences in trim elevator deflection result in the simulated airplane having about 1° more deflection from trim condition than the real airplane.

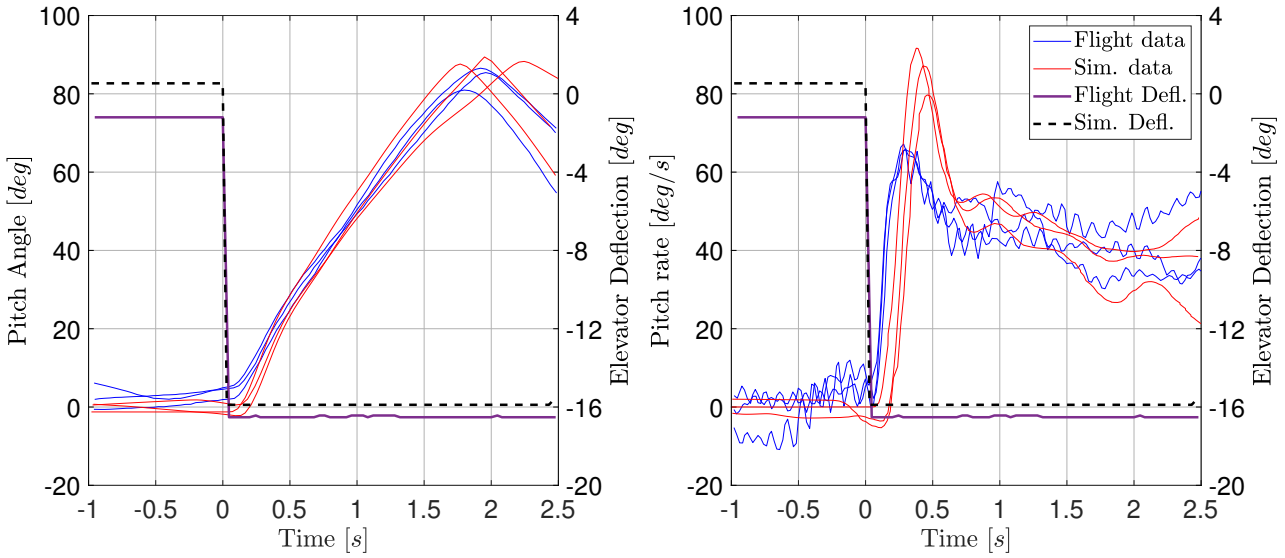


Figure 12 – Pitch up response to step deflection of Elevator

Figure 13 shows notable differences between the roll response of the simulation model and real airplane. Both the roll rate and initial roll acceleration are under-predicted by the simulation model. This can indicate that the aileron effectiveness is underpredicted, the roll damping is overpredicted, the roll inertia of the model is higher, or some combination of the three. On top of this, differences in trim deflection also appear for roll. This time, for a right roll, the result is roughly 1.5° more aileron deflection compared to the simulation model.

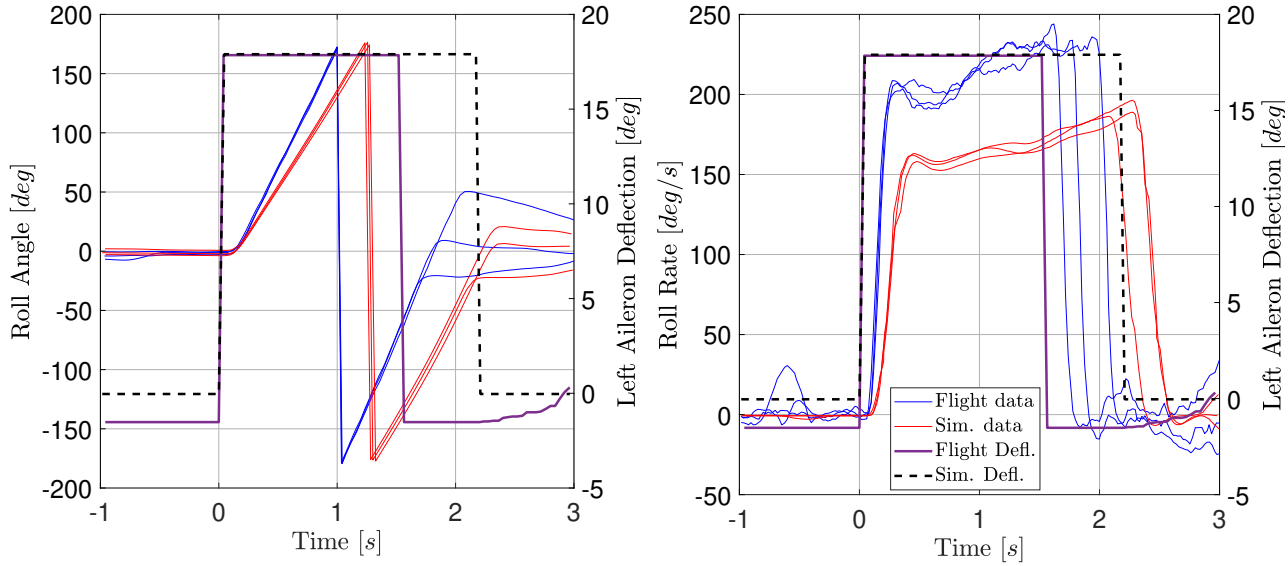


Figure 13 – Roll right response to step deflection of Ailerons

The performance metrics for these maneuvers are summarized in Table 1. For the pitch axis, it is notable that the time to angle peak matches real flights well, but the time to rate peak does not. It appears that when performing maneouvers in the simulator, there is a slight delay between the command signal being sent and the aircraft responding. This could be due to a slightly out of trim aircraft, communication delays between autopilot and simulator, or the simulator's delay in processing changes in the flight dynamics. For the roll axis, the differences between simulated and real flights depend on the direction of the roll. This is most likely due to the propeller model producing inaccurate reaction torque in the simulator, although innacuracies in the modelling of the asymmetric aileron deflection could also be of importance.

Table 1 – Manual Maneuvers Performance Metrics

Maneuver	**Environment**	**Average** T_{AP} $[s]$	**Average** T_{RP} $[s]$	**Average** RP $[deg/s]$
Pitch Up	Real	1.892	0.289	66
	Simulated	1.988	0.421	86
	Difference $[\%]$	5	46	30
Roll Right	Real	0.999	0.318	206
	Simulated	1.254	0.456	161
	Difference $[\%]$	26	44	22
Roll Left	Real	1.052	0.413	-191
	Simulated	1.504	0.490	-134
	Difference $[\%]$	43	19	30

4.1.2 Autotuning

Due to time constraints, 61 autotuning runs were flown in the simulation, while only 11 were flown with the real airplane. The results for the Pitch loop are shown in Figure 14. The proportional gains show good agreement between the two environments. The difference in FF and I terms is consistent with the hypothesis that the real aircraft has less elevator authority than the simulator model. The slightly higher D gain resulting from real flight testing may indicate that the real aircraft has lower pitch damping than the simulation model. However, given the low number of real flight samples and the large spread in D gains resulting from autotuning, this can only be interpreted as a possibility.

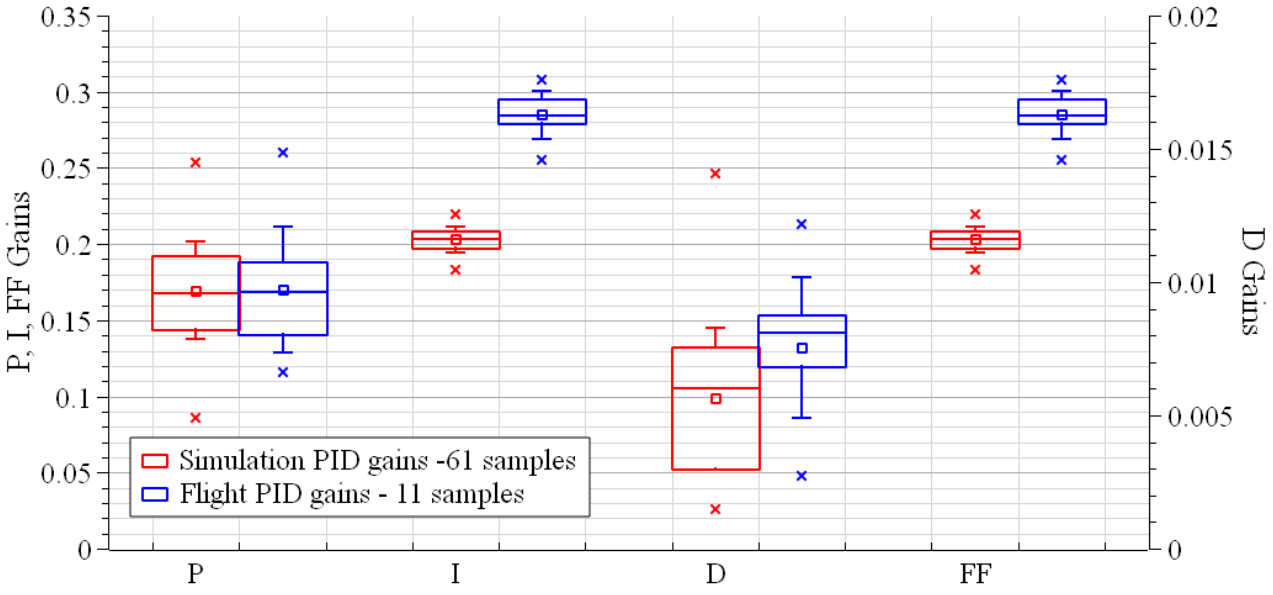


Figure 14 – Pitch PIDFF autotuning results from simulated and real flights

The results for the Roll loop are shown in Figure 15. All simulation-obtained gains are higher than the real-flight values. This discrepancy is consistent with the previously observed differences in the responses to manual inputs of the model and the real airplane. An overestimation of roll damping and an underestimation of aileron effectiveness would naturally drive the autotuning algorithm to output higher P, I, and FF gains.

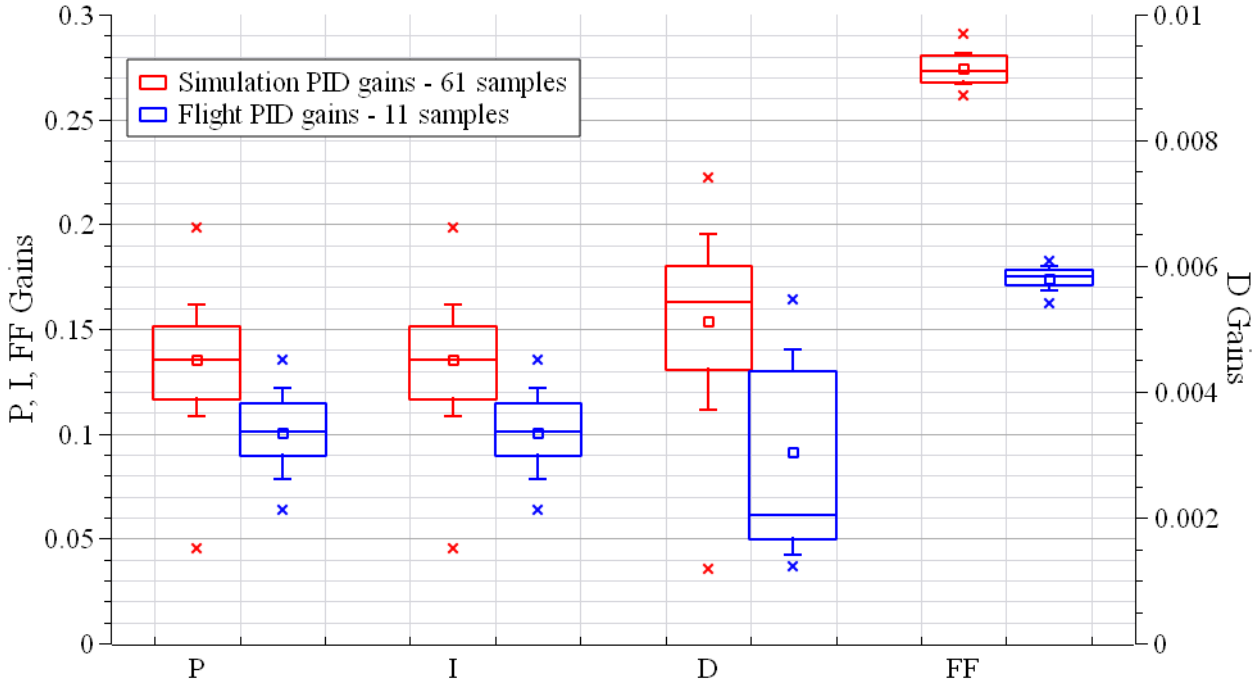


Figure 15 – Roll PIDFF autotuning results from simulated and real flights

A summary of the selected gains and their relative differences is presented in Table 2. The flight testing gain set is denoted *FG* while the simulation gain set is denoted *SG*. For the gains originating from the simulation, either the mean or the median of the dataset was selected as representative, depending on the data distribution. However, due to time constraints, the selected values for flight testing gains were based only on a brief analysis of the data. Hence, some values, such as the Pitch P gain, do not represent either the mean or the median of the dataset. While this is not ideal for comparisons, the values chosen are still within the interquartile range of their respective datasets.

Table 2 – Selected PIDFF gains from simulation and flight testing

Control Loop	Origin	P	I	D	FF
Pitch	Flight Testing (FG)	0.140	0.285	0.00969	0.285
	Simulation (SG)	0.170	0.203	0.0067	0.203
	Difference [%]	18	40	45	40
Roll	Flight Testing (FG)	0.109	0.109	0.0017	0.174
	Simulation (SG)	0.131	0.131	0.005	0.274
	Difference [%]	17	17	66	36

While the differences between the selected simulation and flight-testing gains are significant, the simulation-derived parameters may still be conservative enough to serve as a safe initial configuration. A 66% difference in the derivative gain, for example, will typically cause a measurable change in controller performance. However, Arduplane's autotuning algorithm selects P, I and D gains two to three times lower than those that would cause oscillations. On top of this, the highest differences are present in the D terms, which can typically be set to zero for initial testing. Too high FF gains can also cause issues such as excessive overshoot and overly violent initial responses to inputs, but will not typically cause self-sustaining oscillations.

4.1.3 Responses to step inputs

The two gain sets from Table 2 were uploaded to the autopilot of the real subscale aircraft and tested. The response of the autopilot's pitch angle controller with the *SG* and *FG* gains is depicted in Figure 16. These step responses show that the simulation-derived gains produce very similar behavior to the gains obtained from real flight testing. This is supported by the pilot feedback from the flight testing day and the performance metrics in Table 3. For both pitch-up and pitch-down maneuvers, the rise time is almost identical between the two gain sets. The largest differences appear in the undershoot, where the SG gains perform better compared to the FG set, likely due to the higher proportional gain.

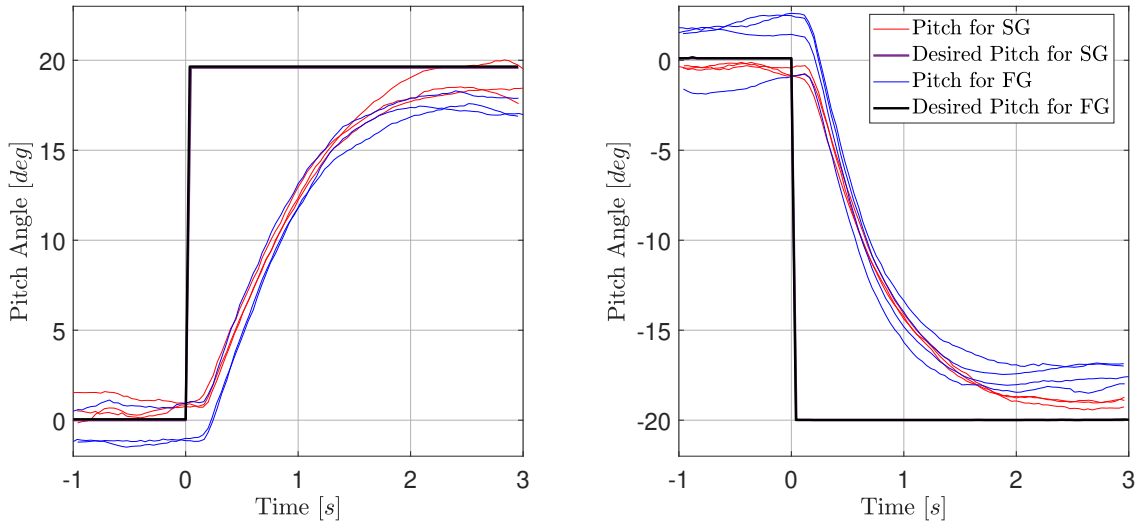


Figure 16 – In-flight pitch angle response to pitch angle step demands for FG and SG gain sets

The same comparison, but this time for the roll controller, is depicted in Figure 17. Here, the step responses show a more noticeable difference between the two configurations, which concurs with the pilot feedback that the aircraft felt slightly more responsive in roll when using the SG gains. The performance metrics in Table 3 support this observation, showing an 18% to 22% faster rise time with the simulation gains compared to the in-flight tuning gains. This is due to the more aggressive proportional and feed-forward gains present in the SG set. On top of this, the higher derivative gain in the SG set leads to almost no overshoot compared to the FG configuration.

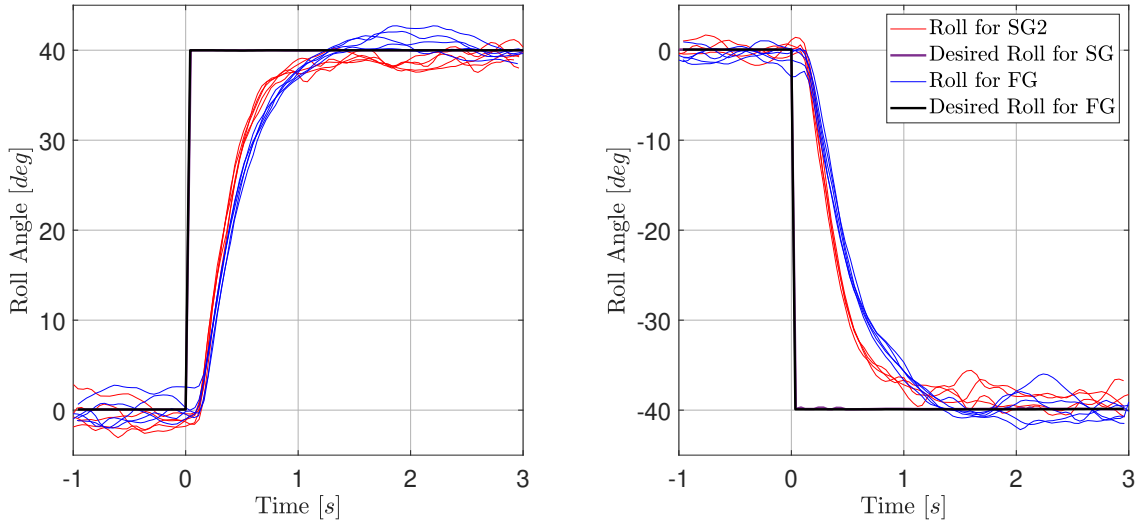


Figure 17 – In-flight roll angle response to roll angle step demands for FG and SG gain sets

Table 3 – SG vs FG Flight Testing Performance

Maneuver	Gain Set	T_R $[s]$	T_T $[s]$	T_S $[s]$	OS $[\%]$	US $[\%]$	AP $[deg]$	$T_{AP}[s]$
Pitch Up	FG	1.203	n/a	n/a	0	11.1	17.8	2.331
	SG	1.197	n/a	n/a	0	5.0	19.0	2.760
	Difference $[\%]$	0.6	n/a	n/a	0	54.8	6.8	18.4
Pitch Down	FG	1.010	n/a	n/a	0	12.4	-17.5	2.104
	SG	1.027	n/a	n/a	0	4.2	-19.2	2.680
	Difference $[\%]$	1.7	n/a	n/a	0	66.3	9.4	27.4
Roll Right	FG	0.711	0.881	0.878	4.8	0	41.9	1.987
	SG	0.580	0.713	0.720	0.8	0	40.3	2.149
	Difference $[\%]$	18.3	19.0	18.1	83.2	0	3.8	8.1
Roll Left	FG	0.824	1.257	1.255	3.8	0	-41.5	2.497
	SG	0.642	1.020	1.024	0.3	0	-40.1	2.650
	Difference $[\%]$	22.0	18.8	18.4	93.2	0	3.4	6.1

It is worth noting that the SG gains perform slightly better on the real aircraft than the FG set, by most performance metrics tracked here. However, this should not be interpreted as the simulation-based tuning method being superior. The ideal outcome for this validation study would be identical performance between simulation derived and flight testing derived gains. Any difference, even one that improves specific metrics like rise time or undershoot, indicates underlying inaccuracies or inconsistencies somewhere in the modeling, simulation, or tuning process.

That being said, despite the inaccuracies in the simulation model and their impact on the resulting controller tune, these subscale flight tests demonstrate that the overall approach produces acceptable, flyable results. Even though the simulation environment yielded a more aggressive tune than the real flights, it did not produce unstable or oscillatory behavior. The subscale testing results validate that using the proposed modeling, simulation, and tuning framework produces a usable initial controller tune for the type of aircraft presented in this paper.

4.2 Large Scale UAV

The results of the subscale validation study provided sufficient confidence in the tuning process to use it for risk mitigation within the scope of one of the F720's flight testing campaigns. After incorporating the improvements described in subsection 3.5, the process was applied to the F720 UAV.

The resulting angle controller gains are presented in Table 4 alongside the standard Arduplane defaults for comparison. As shown, the stock parameters are very low for an airframe of this scale, with some defaults being over ten times lower than the tuned values. If these default gains were engaged in flight, the UAV's response to command inputs would be dangerously sluggish, hindering the autopilot's ability to control the aircraft's attitude. Note that the Derivative gains, marked with a star, were not used for the reasons outlined in subsection 3.5.

Table 4 – Comparison of F720 Controller Gains Against Arduplane Defaults

Control Loop	Configuration	P	I	D	FF
Pitch	**Selected Gain**	**0.51**	**1.00**	**0 (0.02*)**	**1.00**
	Arduplane Default	0.04	0.15	0	0.345
	Factor over Default	12.8×	6.7×	n/a	2.9×
Roll	**Selected Gain**	**0.63**	**0.63**	**0 (0.04*)**	**1.05**
	Arduplane Default	0.08	0.15	0	0.345
	Factor over Default	7.9×	4.2×	n/a	3.0×

The flight testing of the F720, was primarily conducted as part of an independent campaign with a separate objective than that of this paper. Dedicated flight windows for gathering data specifically for this study were limited. However, during available spare testing time, a set of closed-loop step responses was executed in ArduPilot's FBWA mode for both the pitch and the roll axis.



Figure 18 – F720 flying during flight testing campaign

Figure 18 was captured during one of the flights during this campaign. Figure 19 depicts the trajectory of the airplane in relation to the airfield for this flight. Except for the take-off and landing, most of the flight time was flown with the autopilot's attitude control loops engaged. Pilot feedback indicated good roll response but slow pitch response in attitude controlled modes. This is due to a deliberate choice to impose conservative time constants and rate limits for the pitch axis to prevent the autopilot from commanding aggressive maneuvers.

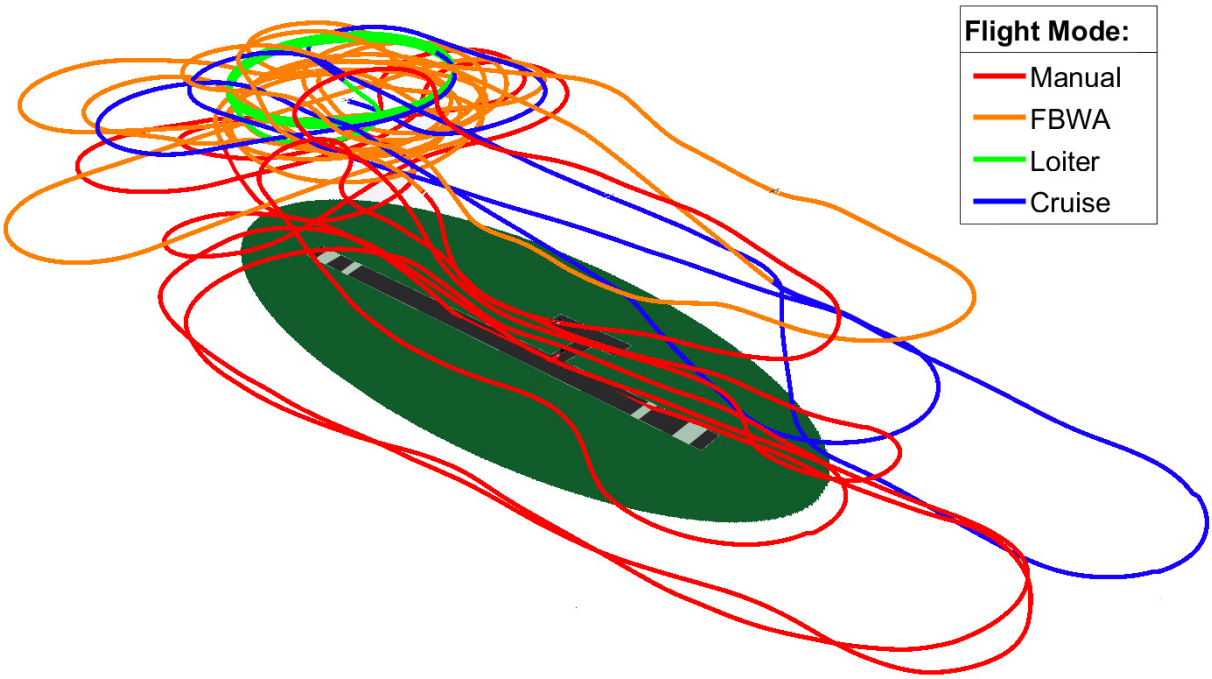


Figure 19 – Trajectory of the F720 and active Autopilot Flight Modes during a test flight

Figure 20 is a short excerpt taken from a section of this flight flown in FBWA. This shows acceptable tracking for pitch and roll commands. Flying with the autopilot in control of attitude proved uneventful and no undesirable behaviors were noted.

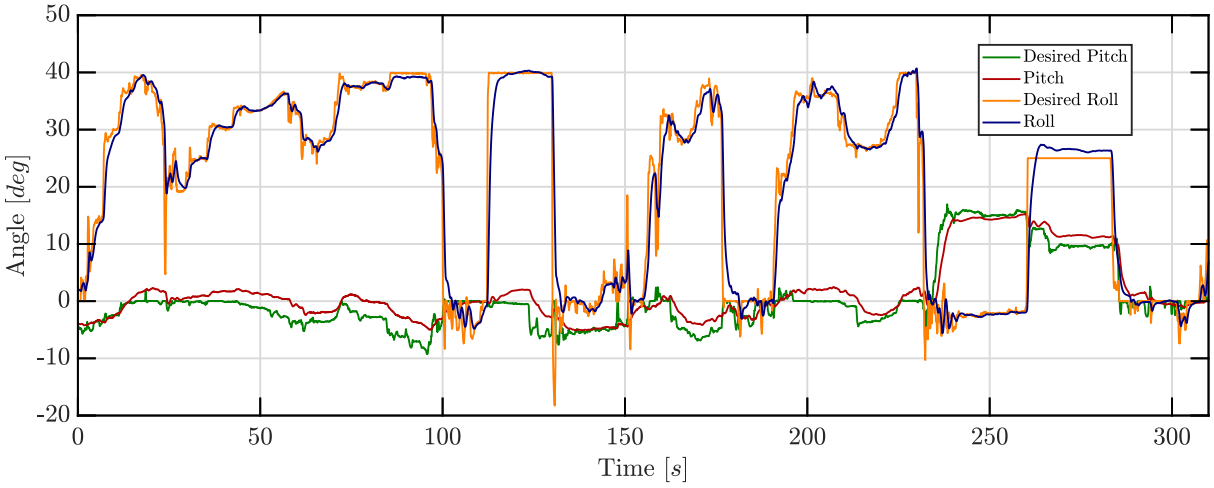


Figure 20 – FBWA Roll and Pitch responses during a flight segment of the F720

Figure 21 depicts the responses of the real F720 and the simulator model to step commands of roll and pitch angle. Both are using the selected gains from Table 4. The simulator model appears to have a faster initial response to the pitch commands, but also exhibits some oscillation in pitch rate. Both the model and the real airplane exhibit undershoot due to loss of airspeed during the commanded climb. In roll, the simulator model responds slower and overshoots the target roll angle, while the real aircraft responds faster and slightly undershoots the target. Table 5 shows that the rise time of a pitch command is slower than that of a roll, confirming the pilot's feedback.

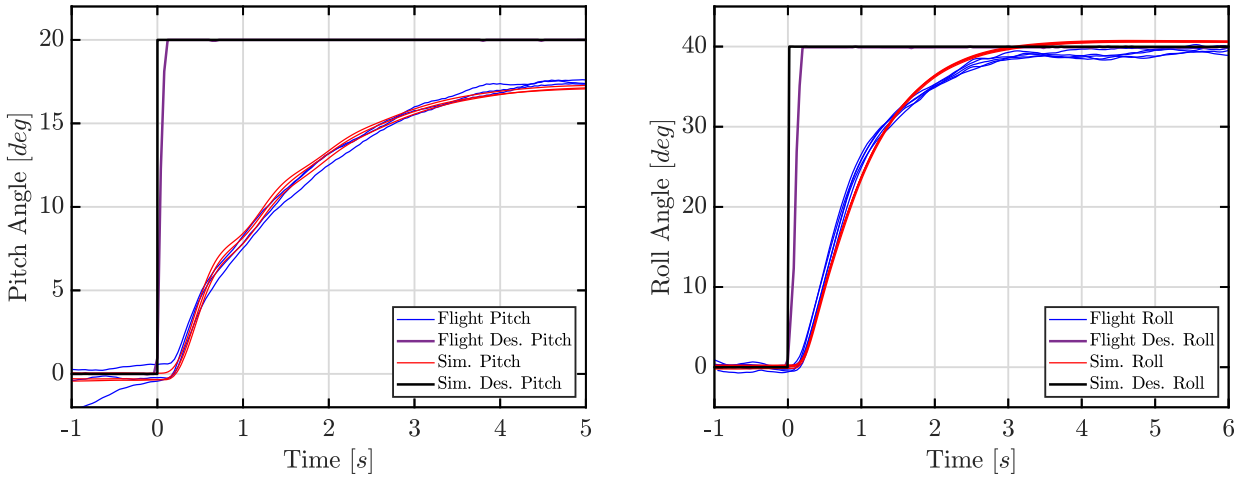


Figure 21 – F720 FBWA Step Responses for Pitch and Roll demands in real and simulated flights

Table 5 – Real Aircraft Step Response Performance (Gains obtained from simulation)

Maneuver	T_R $[s]$	T_T $[s]$	T_S $[s]$	OS $[\%]$	US $[\%]$	AP $[deg]$	T_{AP} $[s]$
Pitch Up	2.852	n/a	n/a	0.0	4.37	17.51	5.827
Roll Right	1.889	3.175	3.176	0.20	0.27	39.85	6.707

5. Discussion

The primary objective of this study is to investigate whether a tuning process built around the X-Plane 12 flight simulator can reduce the risks associated with in-flight tuning of UAV control systems. For the subscale UAV, simulation-derived gains were compared with gains obtained through in-flight tuning, resulting in only minor differences in closed-loop behaviour. For the full-scale UAV, no in-flight tuning was performed, and the aircraft was flown directly using simulation-derived gains, operating uneventfully with stable closed-loop response.

The two aircraft models used in this study are not perfect. Notably, our results indicate that X-Plane 12 underestimates the roll response and slightly overestimates the pitch response, at least for the subscale aircraft. While the underestimation of roll response was also noted in the works of Johnson [5] and Nguyen et al. [13], those studies also reported good agreement in pitch behavior. The observed discrepancy in pitch response in this work is likely due to our subscale aircraft having an unusually flexible tail structure, which reduces elevator effectiveness due to aeroelastic effects.

Despite these modelling inaccuracies, the simulation-derived controller gains still produced acceptable results on both aircraft. This can be primarily attributed to two factors. First, both aircraft considered in this study are stable in pitch and roll and exhibit relatively low agility compared to similarly sized aircraft. This results in reduced sensitivity to controller tuning variations in closed-loop operation compared to aerobatic or high-maneuverability airframes. Second, ArduPilot's Autotune feature increases gains until oscillatory behavior is observed and then applies a conservative scaling factor to obtain final controller gains. Together, these airframe characteristics and the chosen tuning procedure increase tolerance to modelling inaccuracies.

From a practical perspective, the results suggest that the proposed simulation-based tuning process can reduce several of the risks associated with initial in-flight tuning of UAV control systems. In this study, this process eliminated the need for initial in-flight controller tuning on the full-scale UAV, thereby reducing first-flight risk and decreasing the number of dedicated tuning flights required. This, in turn, reduced overall flight campaign cost and complexity. Furthermore, the use of X-Plane 12 allowed for rapid development of the aircraft model, significantly reducing the engineering time required to obtain an initial set of controller gains suitable for flight testing.

A number of limitations should be considered when interpreting the results of this study. First, only two aircraft with similar airframe layouts were evaluated, limiting the generality of the findings to other airframe types. In addition, only a single communication and simulation architecture was used, and the effects of communication delays within this framework were not explicitly quantified. The study also did not include an in-depth model validation effort. Instead, model fidelity was assessed qualitatively through a limited number of open-loop and closed-loop maneuvers, and quantitatively, through non-exhaustive performance metrics. Furthermore, ArduPilot's Autotuning algorithm was used for expediency and to remove subjectivity from the tuning process. The controller gains presented should be treated as a rough initial tune, not as a fine-tuned solution.

Additional limitations stem from the use of X-Plane 12 as the simulation environment. As a closed-source product, the internal modelling assumptions are difficult to fully identify, and the sources of modelling inaccuracies are therefore difficult to isolate, despite the diagnostic tools provided. Moreover, X-Plane 12 was not designed as an engineering or scientific tool. Flight dynamics model calculations and communication delays may vary depending on computational hardware and system configuration. Finally, while ArduPilot is open-source and reasonably well documented, the results presented in this work are only valid for ArduPlane versions 4.4.4 (used on the subscale aircraft) and 4.6.3 (used on the full-scale aircraft), and future changes to the control architecture may affect the applicability of the findings.

6. Conclusions

This study investigated the feasibility of using X-Plane 12 as a low-cost risk mitigation tool for performing simulation-based autopilot tuning for fixed-wing UAVs.

A modeling, simulation and tuning process was developed and validated on a subscale UAV before being demonstrated on a full-scale UAV. Flight testing of the subscale aircraft revealed that using simulation-based tuning resulted in only minor differences in closed-loop behaviour compared to traditional in-flight tuning. When the same process was deployed on the full-scale UAV, the resulting controller parameters led to stable flight behavior and, for our application, eliminated the need for initial in-flight tuning.

These findings demonstrate that using X-Plane 12 for simulation-based tuning of conventional, stable fixed-wing UAVs can be used to reduce or eliminate flight time dedicated for initial control loop tuning, potentially reducing operational risk and cost of UAV development.

7. Contact Author Email Address

The contact author is Cezar-Victor Borugă, mailto: cezar-victor.boruga@liu.se

8. Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] Ardupilot Development Team. Arduplane roll, pitch and yaw controller tuning, 2026.

[2] Leong HI. Development of a 6dof nonlinear simulation model enhanced with fine tuning procedures. Master's thesis, University of Kansas, Lawrence, KS, USA, 2008.

[3] Raymer D. *Aircraft Design: A Conceptual Approach, 6th Edition*. American Institute of Aeronautics Astronautics, 2018.

[4] Roskam J. *Airplane Design, Part VI: Preliminary Determination of Aerodynamic, Thrust and Performance Characteristics*. DARcorporation, 1987.

[5] Johnson CP. Development of a software in the loop simulation approach for risk mitigation in unmanned aerial system development. Master's thesis, Oklahoma State University, Stillwater, OK, USA, 2020.

[6] Prameswari C. Simulation model development of a subscale fighter demonstrator: Aerodynamic database generation and propulsion modeling. Master's thesis, Linköping, Sweden, 2017.

[7] PX4 Autopilot Development Team. Px4 autopilot auto-tuning, 2024.

[8] Laminar Research. *X-Plane 12 Plane Maker Manual*. Laminar Research, 2022. Accessed: June 2026.

[9] AirfoilTools. Airfoil tools database.

[10] Torenbeek E. *Synthesis of Subsonic Airplane Design*. Delft University Press, 1982.

[11] Nicolai LM and Carichner WM. *Fundamentals of Aircraft and Airship Design: Volume I*. American Institute of Aeronautics and Astronautics, 2010.

[12] Stinton D. *The Design of the Aeroplane*. Collins, London, UK, 1983.

[13] Nguyen LV, Tun W, Nguyen NV, Tyan M, Kim S and Lee JW. Evaluation on x-plane simulator using flight test data of light aircraft kla-100 scaled model. In *Proceedings of the KSAS Fall Conference*, 2016.