Evaluation and Analysis of ArduPilot Automatic Tuning Algorithm for the Roll Tracking Controller of a Small UAS

Justin J. Matt*, Harold P. Flanagan† and Haiyang Chao‡
University of Kansas, Lawrence, KS 66045, USA

PID controllers are widely used for UAS flight control, and tuning of the control law is crucial to the success of a flight mission. In this paper, the automatic tuning process onboard the popular ArduPilot autopilot was examined, analyzed, and validated in both simulation and UAS flight test, considering both time domain and frequency domain controller specifications. The autotune algorithm was first implemented in MATLAB and simulated with a PID and PD controller using identified KHawk UAS lateral dynamic models. The results were further compared with fight test data. Frequency domain analysis showed the effectiveness of the autotune algorithm, as most of the autotuned controllers, except those with very aggressive gain sets, demonstrated performance comparable to controllers designed using a model-based approach. Specifically, the autotuned controllers showed high stability margins and desirable disturbance rejection capability.

I. Introduction

NOWADAYS, unmanned aircraft systems (UAS) are commonplace in military and civil applications, and many of these aircraft use simple proportional-integral-derivate (PID) controllers for flight stabilization and autonomous path following. A critical part of flight controller design is gain determination, or tuning. Controller tuning during a flight test can be expensive, dangerous, and time-consuming. Traditional tuning approaches are mostly model based, which require significant efforts in aircraft model identification through aerodynamic modeling, wind tunnel testing, and flight testing. In contrast, the introduction of low-cost autopilots and UAS platforms made it possible to perform in-flight control gain tuning based on real-time observations. The widely used ArduPilot open source software suite provided an automated tuning process for the determination of inner loop controller gains for any small UAS, commonly powered by a Pixhawk autopilot. The introduction of automated gain tuning greatly simplifies the flight preparation process for many UAS users.

Though uncommon in aerospace applications, automatic PID tuning has been used in other industrial systems, including liquid level control systems [1], urban rail systems [2], neonatal incubators [3], and servomotors [4]. Neural networks [1, 4], fuzzy logic controllers [5, 6], relay feedback controllers [3, 7], and unfalsified control theory [8] have all been used as methodologies to achieve automated and real-time PID tuning. Unfalsified control theory is a particularly useful method as knowledge of the system model is not required. Instead, the controller adapts based solely on measurement data, adjusting the control law such that the closed-loop system response satisfies provided controller specifications [9]. One application of unfalsified control is the automatic tuning of a PID controller based on the measured system response [10]. The ArduPilot automatic tuning algorithm aligns with the principle of unfalsified control as it adjusts the controller on the condition of a single time-domain performance metric, the percent overshoot (PO) of aircraft angular rates.

This paper analyzes the ArduPilot automatic tuning process, or "autotune," for the roll tracking controller through simulation and flight test data. The algorithm is designed to automatically tune aircraft inner-loop controller gains during flight tests. The main contributions of this paper are the simulation and flight test validation of the ArduPilot autotune algorithm and the evaluation and analysis of the resulting controller performance using identified UAS dynamic models. Additionally, the autotuned controller is compared with controllers designed using model-based approaches [11].

This paper is organized as follows. The roll tracking controller structure is discussed in Section II. The ArduPilot automatic tuning algorithm and its implementation are presented in Section III. The UAS platforms and lateral dynamic models used for simulation and flight validation are introduced in Section IV. Simulation results, including comparisons to flight test data and controller performance analysis in the time and frequency domains, are discussed in Section V.

*M.S. Student, Department of Aerospace Engineering, justinjmatt@ku.edu
†Ph.D Candidate, Department of Aerospace Engineering, h682f304@ku.edu
‡Associate Professor, Department of Aerospace Engineering, chaohaiyang@ku.edu

II. Roll Tracking Controller Structure

The objective of the ArduPilot autotune program is to determine the gains for the two inner-loop attitude tracking controllers. In this paper, two roll tracking controller structures are analyzed. The ArduPilot roll controller is introduced first, then, a classical PD controller is presented for comparison with controllers designed using model-based approaches.

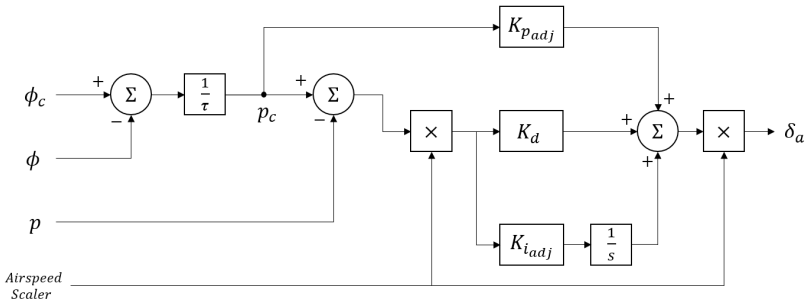

Fig. 1 ArduPilot PID controller structure.

The ArduPilot roll controller is shown in Fig. 1, where ϕ indicates roll angle, p indicates roll rate, and a subscript c indicates a commanded value, rather than a measured value. The controller follows a conventional PID structure, with two concepts of note that deviate from this. First, the error is multiplied by an "airspeed scaler," which decreases the output deflection as the airspeed increases. This scaler is not typically used in conventional roll tracking controllers for UASs [14]. Second, the PID gains are based on the commanded roll rate and roll rate error, rather than roll angle. Because of this, the commanded roll rate and PID gains are adjusted using a time constant, τ, shown in Eqs. (1-3), which represents the time taken to achieve a commanded roll angle.

$$K_{p_{adj}} = K_p\tau - K_i\tau^2 - K_d \tag{1}$$

$$K_{i_{adj}} = K_i\tau \tag{2}$$

$$p_c = (\phi_c - \phi)/\tau \tag{3}$$

This adjustment ensures certain scaling so that the gains can be tuned using standard PID logic. Additionally, the controller accounts for the adverse effects of integrator wind-up by constraining the change in the integrator term when the control surfaces are saturated.

A second approach was taken using a classical PD controller structure, shown in Fig. 2. This was done for comparison with a model-based roll controller previously developed to optimize multiple objectives, including disturbance rejection bandwidth (DRB), gain margin (GM), and phase margin (PM) [11].

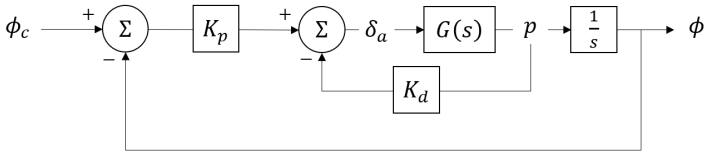

Fig. 2 Classical PD controller structure [12].

III. ArduPilot Automatic Tuning Algorithm

The ArduPilot automatic tuning algorithm follows a simple algorithm based on the overshoot of the roll rate or pitch rate response. ArduPilot provides an autotune level for users to adjust the aggressiveness of the automatically tuned controller. The autotune level ranges from one to eleven, where higher levels represent a more aggressive controller. Level 6 is the default value recommended for most UASs. For the roll controller, the autotune level determines three parameters: the roll angle time constant, τ, D-gain ratio, D_{ratio}, and maximum roll rate, p_{max}, which are shown in Table 1. As the level increases, τ decreases and D_{ratio} and p_{max} increase. In practice, increasing the autotune level usually results in a faster but less stable response.

Table 1 Autotune level parameters.

Level	1	2	3	4	5	6	7	8	9	10	11
τ	0.70	0.65	0.60	0.55	0.50	0.45	0.40	0.30	0.20	0.10	0.05
D_{ratio}	0.050	0.055	0.060	0.065	0.070	0.075	0.080	0.085	0.090	0.095	0.010
p_{max}	20	30	40	50	60	75	90	120	160	210	300

The autotune algorithm follows the steps shown below and in Fig. 3 based on time domain specifications. The process updates at 50 Hz.

1) Ensure the control surface command is unsaturated, otherwise terminate autotuning.
2) Ensure controller is demanding or achieving greater than 80% of p_{max}, otherwise terminate autotuning
3) If the roll rate measurement overshoots for more than 0.1 seconds, decrease K_p by 8%. Else, if the roll rate measurement undershoots for more than 0.2 seconds, increase K_p by 5%.
4) Update K_i and K_d based on the rule of thumb from Eqs. (4) and (5)

$$K_d = K_p D_{ratio} \tag{4}$$

$$K_i = 0.5 \cdot K_d/\tau \tag{5}$$

where K_d is the derivative gain, K_i is the integrator gain, and D_{ratio} and τ are specified by the autotune level.

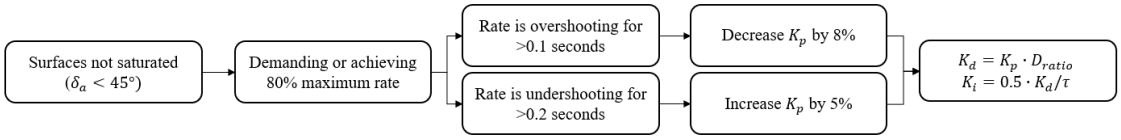

Fig. 3 Autotune algorithm structure

The autotune process begins from a set of pre-adjusted gains, and the controller automatically adjusts the values iteratively based on roll angle and roll rate measurements. During flight, ArduPilot recommends 20 to 30 rapid, oscillatory roll and pitch maneuvers in the attitude tracking mode to achieve successful autotuning. In other words, the pilot must demand aggressive roll or pitch angles through rapid movements of the RC joystick. This ensures that the 80% maximum rate constraint is met and that the algorithm has enough responses to converge to a desirable gain set.

IV. UAS Platforms and Lateral Dynamic Models

The lateral dynamic model of the KHawk-55-1 UAS is used in this paper for simulation validation. KHawk-55-1 is a flying-wing UAS developed by the Cooperative Unmanned Systems Laboratory (CUSL) at the University of Kansas. It has a wingspan of 55 inches, gross takeoff weight of 5.75 lbs, and the cruise airspeed is 18 m/s. A single electric brushless motor powers the UAS, producing 1.5-2.5 lb thrust. The only control surfaces on the UAS are left and right elevons, which perform all longitudinal and lateral flight maneuvers. The KHawk-55-1 UAS is powered by a Paparazzi TWOG autopilot board together with a Microstrain GX3-25 IMU and a Ublox LEA 6H GPS receiver. Lateral dynamic models of this UAS were identified previously [13] and are used for simulation. Both a first order and a high order aileron-to-roll-rate model were used [13].

$$\frac{p}{\delta_a} = \frac{297.5e^{-0.131s}}{s + 28.46} \tag{6}$$

$$\frac{p}{\delta_a} = \frac{143.3s(s^2 + 2(0.23)4.16s + 4.16^2)e^{-0.114s}}{(s - \frac{1}{9.98})(s + \frac{1}{0.103})(s^2 + 2(0.22)5.05s + 5.05^2)} \tag{7}$$

Additionally, flight test data from the CUSL KHawk-55-3 was analyzed to compare between experimental and simulation results. This UAS is a newer system built from the same airframe as the KHawk-55-1. However, it is powered by a Pixhawk autopilot and has slightly different weight properties and configuration. Both UASs are shown in Fig. 4.

Fig. 4 CUSL KHawk-55-1 (left) and CUSL KHawk-55-3 (right).

V. Controller Evaluation, Analysis, and Experimental Validation

The ArduPilot autotune algorithm is first simulated using the identified dynamic models for the KHawk-55-1 UAS. Experimental validation is performed using flight test data from the similar KHawk-55-3 UAS. The performance of the autotuned controller is further compared with a classical PD controller designed using a model-based approach with multi-objective optimization. Both time domain and frequency domain controller specifications are compared and analyzed to evaluate the autotuned controller performance.

The controller was modeled and simulated in MATLAB using the structure shown in Fig. 5. The simulation follows the below steps.

- From Eq. (3), the roll controller calculates the demanded roll rate, which is input to the autotune program.
- Based on the autotune level, the program adjusts the gains, following the algorithm shown in Fig. 3.
- The controller calculates the adjusted gains from Eqs. (1) and (2) and updates the aileron demand, δ_a, which is sent to the dynamic model.
- The dynamic model simulates the UAS response, outputting the measured roll angle and roll rate.

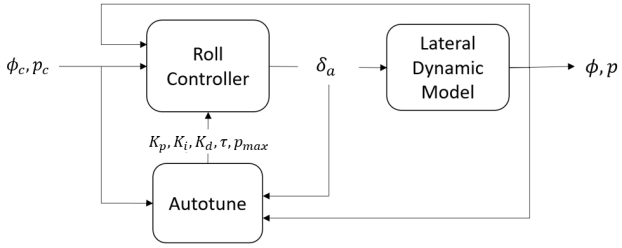

Fig. 5 Simulation structure.

A. Simulation Results

The UAS response was simulated at autotune Levels 4, 6, and 8, using the PID controller for the first order model. From Fig. 6, it can be observed that the algorithm successfully improves the system response over time for each of the three autotune levels. The higher levels result in a faster response but less damping of oscillations. The recommended Level 6 is a compromise between these parameters that provides well-rounded performance.

Figure 7 shows that the program successfully converged to a set of gains for each of the three observed autotune levels. A higher level corresponded with higher, more aggressive gains. Once a desirable response was achieved, the algorithm continually adjusts the controller, which may cause the gains to oscillate around a final value.

One observation from the autotune simulation is that roll rate overshoot is sometimes incorrectly identified. This is caused by delays in the system response. From Fig. 7, at 140 seconds, the Level 6 autotune increases the P-gain after it had settled for some time, indicating that the roll rate must have been undershooting. However, it can be observed from Fig. 8 that the roll rate response is overshooting, and that the delay is what is causing an initial undershoot. Though all aircraft behave differently, the autotune algorithm could be improved by implementing a "catch-all" value for system delay. This will increase the program's ability to properly identify overshoot.

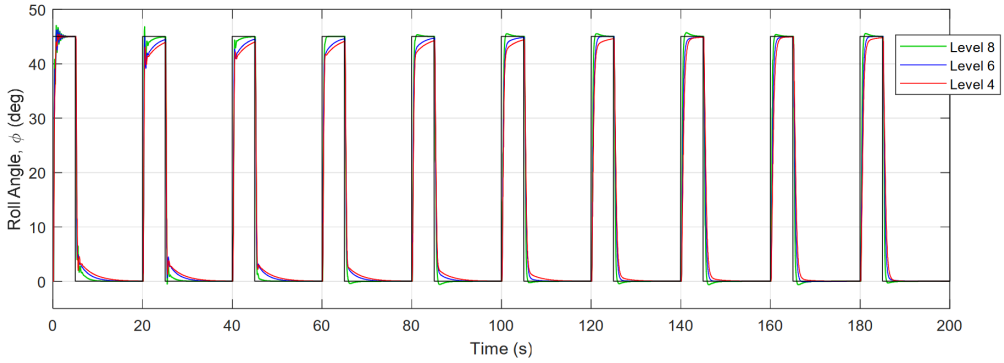

Fig. 6 Roll tracking response using first order dynamic model at different autotune levels.

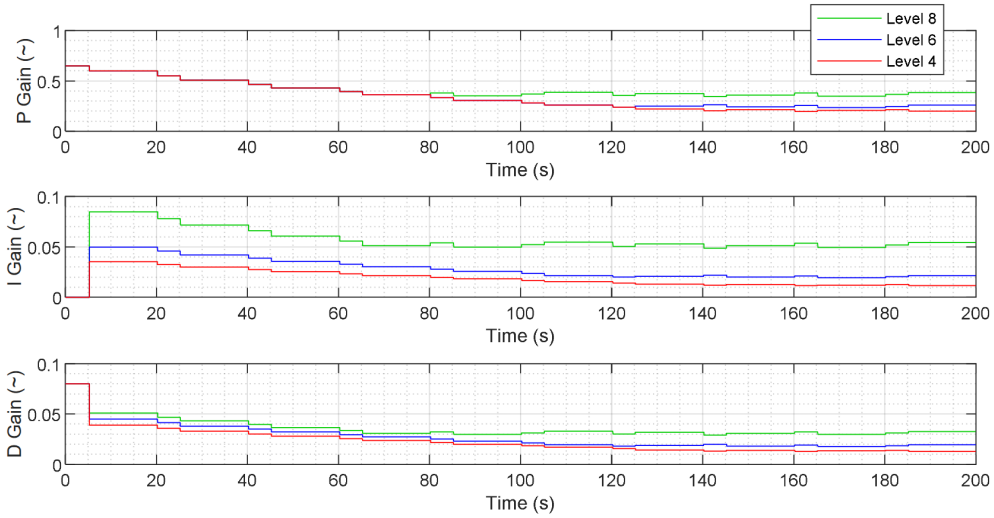

Fig. 7 PID gain response using first order dynamic model at different autotune levels.

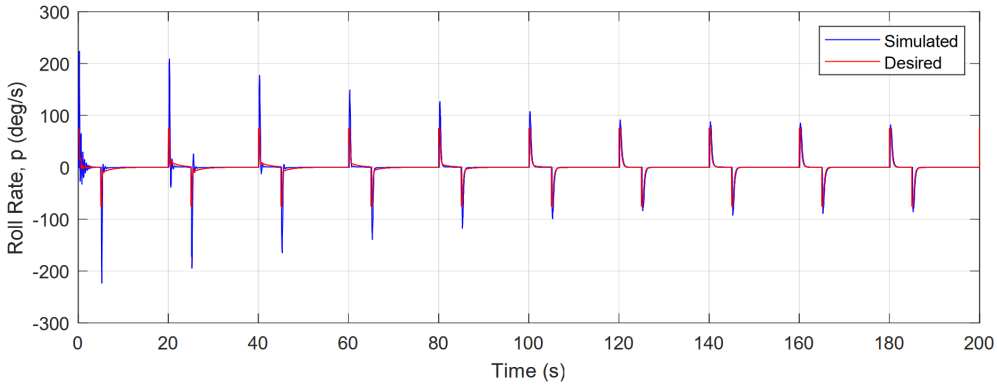

Fig. 8 Roll rate response using first order dynamic model (Level 6).

This issue is more prevalent when the response of the high order model is simulated. Figure 9 shows the P-gain and roll rate response using the high order model at autotune Level 6. At 100 seconds, the P-gain increases, indicating the measured rate response was undershooting. However, inspecting the entire response at this time shows the rate is overshooting, but with some delay. This misidentification results in higher gains and a less stable roll angle response, seen in in Fig. 10. Despite this, the algorithm still converges to a controller with an improved response.

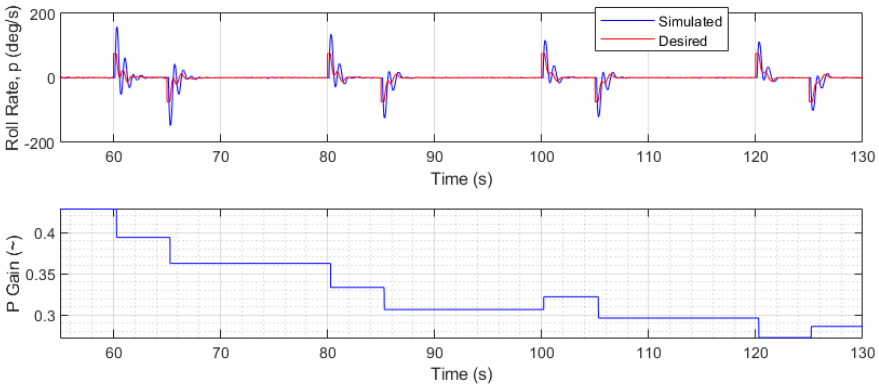

Fig. 9 Roll rate response and autotuned gains using high order dynamic model (Level 6).

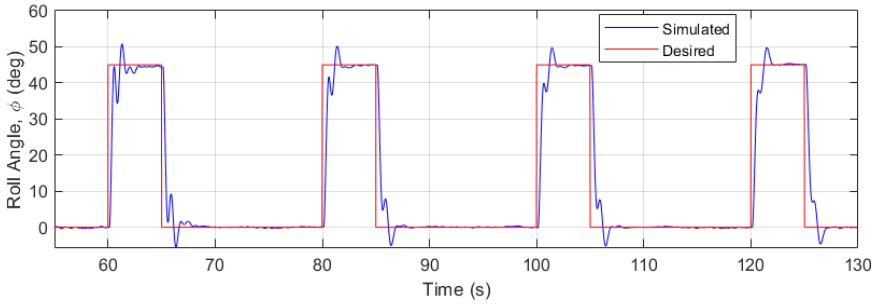

Fig. 10 Roll tracking response using high order dynamic model (Level 6).

B. Experimental Results and Validation

The simulation of the PID controller was compared to real flight data. The KHawk-55-3 UAS was used for flight testing, which is a newer UAS built from the same airframe as the KHawk-55-1 and expected to have similar flight dynamics. Figure 11 shows flight data from the KHawk-55-3 while flying in autotune mode. The response and controller adaptation during the first set of maneuvers (760-790 s) is comparable to the simulated response, approaching a P-gain of about 0.45. However, during the second set of maneuvers the tracking response is poor. From 800-820 seconds, the UAS has a sudden increase in ground speed caused by changing direction and flying with a tailwind. From this observation, it is recommended that pilots attempt to maintain cruise speed during autotuning. During the third set of maneuvers, the algorithm begins approaching a P-gain around 0.6.

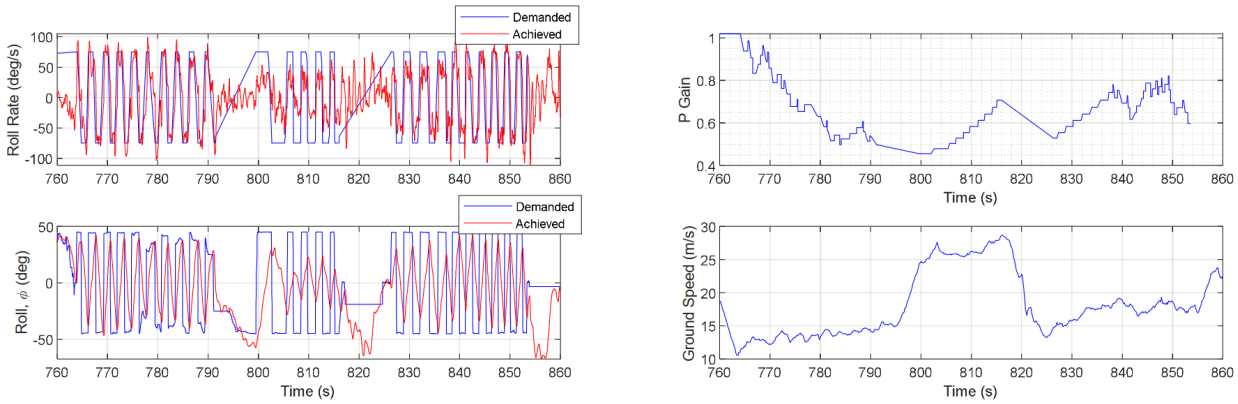

Fig. 11 Roll rate response and autotuned gains during flight test.

C. Comparison with Optimized Controller

Next, the PD controller was implemented for comparison with model-based controller design. The results were simulated for the high order and low order models at different autotune levels. The autotune program successfully damped out oscillations and converged to a controller with a smooth response. The misidentification of roll rate overshoot was not present during the autotuning procedure for this controller. This is likely because the removal of the integrator term results in a faster rise time, so, the measured roll rate will not "undershoot" for the 0.2 seconds required to decrease the P-gain. The roll angle, roll rate, and P-gain responses are shown in Fig. 12, using the first order model and autotune Level 6.

The gains set by the autotune were compared to a controller previously optimized to maximize disturbance rejection bandwidth and gain/phase stability [11], shown in Table 2. The results from the most aggressive automatic tuning, Level 8, were closest to the optimized controller.

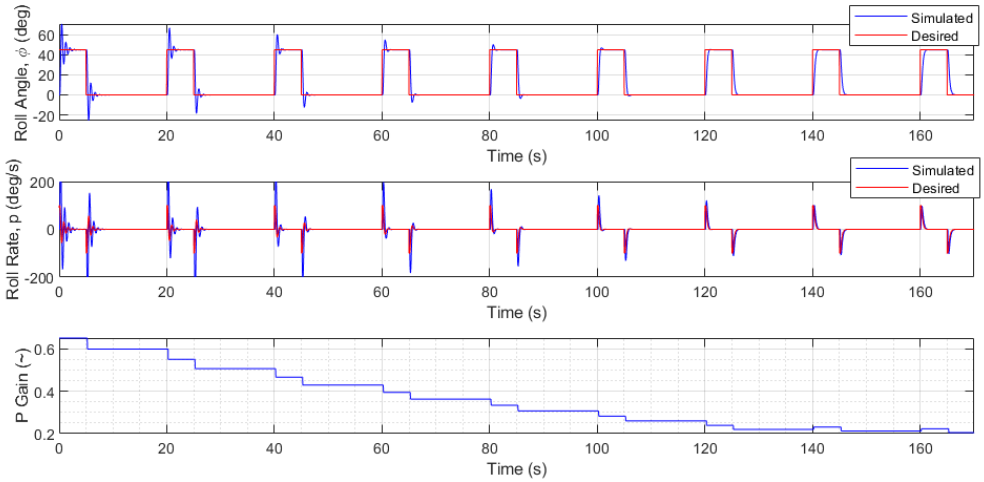

Fig. 12 PD controller tracking response using first order dynamic model (Level 6).

Table 2 Comparison between the autotuned gain set and the model based controller gain set [11, 13].

	Level 4		Level 6		Level 8		Multi-Objective Optimized Controller	
	1st Order	High Order	1st Order	High Order	1st Order	High Order	1st Order	High Order
K_p	0.19	0.23	0.23	0.29	0.32	0.42	0.48	0.42
K_d	0.012	0.015	0.017	0.022	0.027	0.036	0.034	0.046

D. Controller Performance Analysis

The obtained controller performance was analyzed using time domain and frequency domain methods. Frequency domain analysis can be used to better predict the aircraft response [14, 15]. Specifically, the gain and phase margins were identified, providing insight into the stability of the system. Disturbance rejection analysis was performed to evaluate the system response to different measurement noises. A frequency sweep was added to the model feedback to determine the disturbance rejection bandwidth and disturbance rejection peak (DRP).

The first order and high order model were each analyzed using both the ArduPilot PID controller and the classical PD controller. The results using the first order model are discussed first, followed by the results from the high order model. Three gain sets were analyzed for each, representing controllers identified by the automatic tuning algorithm at Level 4, 6, and 8. The gain sets are shown in Table 3 and Table 4.

The controller specifications shown in Table 5 are selected based on [11]. For a small UAS, rise times are generally low as a fast response is desired. The gain and phase margins relate to the stability of the system, while the DRB and DRP show the UASs ability to reject disturbances.

Table 3 Determined gain sets using first order dynamic model.

	PD Controller		PID Controller		
Autotune Level	K_p	K_d	K_p	K_i	K_d
Level 4	0.19	0.012	0.21	0.004	0.014
Level 6	0.23	0.017	0.25	0.004	0.019
Level 8	0.32	0.027	0.38	0.006	0.032

Table 4 Determined gain sets using high order dynamic model.

	PD Controller		PID Controller		
Autotune Level	K_p	K_d	K_p	K_i	K_d
Level 4	0.23	0.015	0.25	0.004	0.016
Level 6	0.29	0.022	0.31	0.005	0.023
Level 8	0.42	0.036	0.46	0.007	0.039

Table 5 Desired controller specifications [11, 14].

Rise Time	0.2 s $< t_{rise} <$ 0.7 s
Percent Overshoot	$PO <$ 10%
Gain Margin	$GM >$ 5.5 dB
Phase Margin	$PM >$ 45 deg
Disturbance Rejection Bandwidth	$DRB >$ 1 rad/s
Disturbance Rejection Peak	$DRP <$ 5.5 dB

The simulation results for the first order models are shown in Table 6 and Table 7. The controllers met all frequency domain specifications, but the more conservative controllers had a slower response than desired. Because a first order model was used, virtually no overshoot was observed. As expected, the higher autotune level results in a faster response, but sacrifices for this with lower gain and phase stability. The disturbance response was determined by adding simulated, variable-frequency disturbances to the controller feedback. Each controller has a low DRP and desirable DRB.

Table 6 Autotuned PD controller performance using first order dynamic model.

	Rise Time (s)	Overshoot (%)	GM (dB)	PM (deg)	DRB (rad/s)	DRP (dB)
Level 4	0.864	0	15.4	78.1	1.43	2.11
Level 6	0.701	0	13.3	76.9	1.62	2.43
Level 8	0.47	0	10.0	73.4	2.02	3.16

Table 7 Autotuned ArduPilot PID controller performance using first order dynamic model.

	Rise Time (s)	Overshoot (%)	GM (dB)	PM (deg)	DRB (rad/s)	DRP (dB)
Level 4	0.77	0.00	17.3	87.2	1.52	2.27
Level 6	0.639	0.02	15.2	91.0	1.71	2.58
Level 8	0.359	0.06	11.0	99.7	2.26	3.70

The results simulating the high order model are shown in Table 8 and Table 9. Except for the Level 8 controllers, the time domain response is slower than desired, and each controller has a high overshoot – near ten percent. This can also be observed from the step response shown in Fig. 13. Additionally, the controllers show steady state error ranging from 2-4%, with the more conservative controllers having higher error. It is evident that a higher autotune level results in a more rapid response, but has lower damping, gain stability, and a higher DRP.

Table 8 Autotuned PD controller performance using high order dynamic model.

	Rise Time (s)	Overshoot (%)	GM (dB)	PM (deg)	DRB (rad/s)	DRP (dB)
Level 4	0.883	9.30	12.4	74.0	1.45	3.70
Level 6	0.812	10.3	10.5	75.1	1.69	4.34
Level 8	0.222	10.2	7.15	46.8	2.14	5.75

Table 9 Autotuned ArduPilot PID controller performance using high order dynamic model.

	Rise Time (s)	Overshoot (%)	GM (dB)	PM (deg)	DRB (rad/s)	DRP (dB)
Level 4	0.858	10.0	16.5	80.5	1.54	4.11
Level 6	0.797	10.7	14.4	84.6	1.76	4.78
Level 8	0.197	10.5	10.6	94.6	2.25	6.58

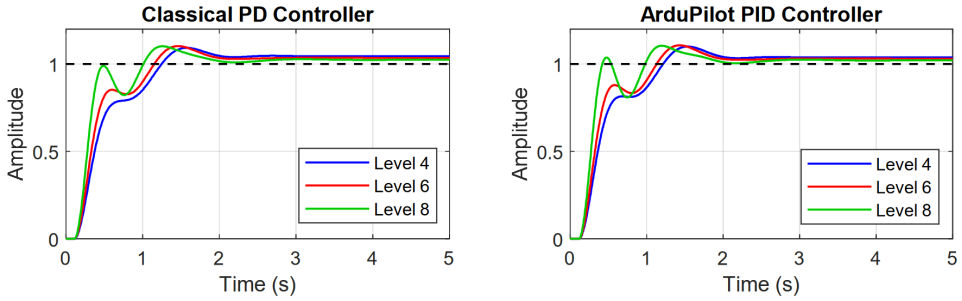

Fig. 13 Simulated step response of the closed loop system using high order dynamic model.

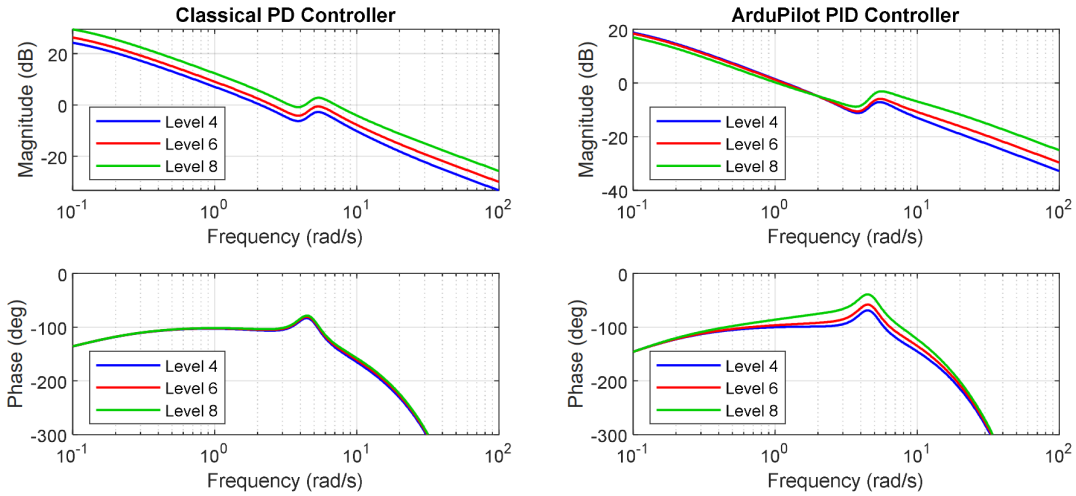

Fig. 14 Simulated broken loop (at actuator) response using high order dynamic model.

Broken loop analysis was used to identify the gain and phase margins. The broken loop response can be obtained by inputting a frequency sweep at the actuator, providing insight to the system behavior when subject to gust-generated actuator disturbances [11, 15]. Additionally, disturbance rejection analysis was performed by adding a frequency sweep at the state feedback to identify the system response to measurement noises [11, 15].

Though the controllers did not meet all time domain specifications, the frequency response was improved. Each controller shows satisfactory gain and phase stability, as seen in Fig. 14, and demonstrates good tracking of low frequency signals while attenuating high frequency noise. Additionally, all the controllers show favorable disturbance rejection, with DRB above the specified value. The Level 8 controllers have a higher DRP than desired, which can be seen as a spike in the plots in Fig. 15.

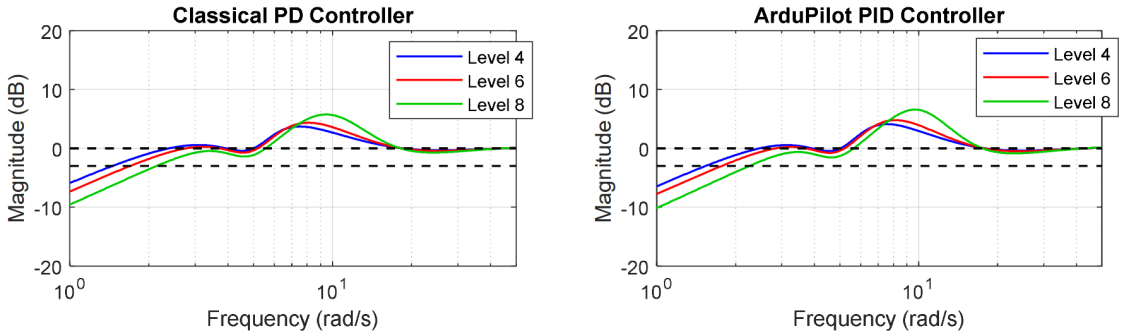

Fig. 15 Simulated disturbance rejection loop response using high order dynamic model.

VI. Conclusion

The paper presents the simulation, flight test evaluation, and analysis of the ArduPilot automatic tuning algorithm. In simulation, the algorithm successfully converges to a gain set with a fast and stable response using first order and high order lateral dynamic models. However, because of delays in the system response, the algorithm sometimes misidentifies whether the response is overshooting or undershooting, and therefore incorrectly adjusts the gains. This problem was not prevalent in the faster PD controller studied. Though all UASs behave differently, it is recommended to add a small-time delay to the autotune algorithm to attempt to alleviate this error.

When analyzed in the time domain, the controllers using the high order model demonstrate high overshoot which is undesirable. However, all the controllers met most of the frequency domain specifications, except for the most aggressive sets of gains, which had a high DRP. Though not all specifications were met, it is evident that the autotune algorithm achieves a response comparable to controllers tuned using model-based design.

Future directions include adjusting or adding to the automatic tuning procedure to better satisfy multiple controller objectives and to correct for anticipated system delay. Additionally, further comparison between simulated and experimental results will provide a better understanding of the interaction between aircraft dynamics and the autotune algorithm. Lastly, a linear state space lateral dynamic model can be used for simulation validation and controller analysis, for a potentially higher correspondence between simulation and flight results.

Acknowledgments

The authors would like to acknowledge the ArduPilot team, specifically Andrew Tridgell, for development and sharing of the ArduPilot autotune function for fixed-wing UASs. The authors are partially supported by USDA-NIFA Grant 2019-67021-28992.

References

[1] Murthy, B. V., Kumar, Y. V. P., and Kumari, U. V. R., "Application of neural networks in process control: Automatic/online tuning of PID controller gains for ±10% disturbance rejection," *IEEE International Conference on Advanced Communication Control and Computing Technologies*, IEEE, Ramanathapuram, India, 2012.

[2] Liu, H., Qian, C., and Liang, Y., "An integrated intelligent control algorithm for urban rail transit automatic train operation system based on FST-PID," *International Conference on Robotics and Automation Engineering*, IEEE, Shanghai, China, 2017.

[3] Pereira, R., and Torrico, B., "New automatic tuning of multivariable PID controller applied to a neonatal incubator," *8th International Conference on Biomedical Engineering and Informatics*, IEEE, Shenyang, China, 2015.

[4] Huang, J., "Automatic tuning of the PID controller for servo systems based on relay feedback," *International Conference on Industrial Electronics, Control and Instrumentation*, IEEE, Nagoyo, Japan, 2000.

[5] Zhang, M., and An, J., "Application of Intelligent Computation to Promote the Automation of Flight Control Design," *International Conference on Intelligent Computation Technology and Automation*, IEEE, Hunan, China, 2008.

[6] Chakraborty, U. K., Bandyopadhyay, R., and Patranabis, D., "A fuzzy-genetic approach for automatic tuning of a PID controller," *23rd International Conference on Information Technology Interfaces*, IEEE, Pula, Croatia, 2001.

[7] Pi-Mira, J., Mateo, E., Sarrate-Estruch, R., and Casin, J. Q., "A three-stage automatic PID tuning system," *European Control Conference*, IEEE, Karlsruhe, Germany, 1999.

[8] Menani, S., and Koivo, H., "Automatic tuning of multivariable controllers with adaptive relay feedback," *35th IEEE Conference on Decision and Control*, IEEE, Kobe, Japan, 1996.

[9] Safonov, M., and Tsao, T., "The unfalsified control concept and learning," *Transactions on Automatic Control*, Vol. 42, No. 6, 1997.

[10] Jun, M., and Safonov, M., "Automatic PID tuning: an application of unfalsified control," *International Symposium on Computer Aided Control System Design*, IEEE, Kohala Coast, HI, USA, 1999.

[11] Flanagan, H., Hagerott, S., and Chao, H., "Model Based Roll Controller Tuning and Frequency Domain Analysis for a Flying-Wing UAS," *International Conference on Unmanned Aircraft Systems*, IEEE, Atlanta, GA, USA, 2019.

[12] Beard, R. W., and McLain, T. W., *Small Unmanned Aircraft: Theory and Practice*, 1st ed., Princeton University Press, Princeton, NJ, 2012.

[13] Flanagan, H., Hagerott, S., and Chao, H., "Model Based Roll Controller Tuning and Analysis for Small UAS in Turbulent Environments," *International Conference on Unmanned Aircraft Systems*, IEEE, Dallas, TX, USA, 2018.

[14] Sanders, F. C., Tischler, M., Berger, T., Berrios, M. G., and Gong, A., "System Identification and Multi-Objective Longitudinal Control Law Design for a Small Fixed-Wing UAV," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Kissimmee, FL, USA, 2018.

[15] Tischler, M., and Kemple, R., *Aircraft and Rotorcraft System identification, Engineering Methods with Flight Test Examples*, 1st ed., AIAA, Reston, VA, USA, 2006.