Control Law Implementation for Rotorcraft UAVs

Judson T. Babcock *
Department of Aeronautics, U.S. Air Force Academy, Colorado, 80841

Commercial off-the-shelf rotorcraft and fixed-wing UAVs offer an inexpensive, modular, and effective platform for implementing control laws for classroom instruction and many real-world missions such as aerial photography, surveying, or even entertainment and motivation toward STEM fields. An optimized control law can aid the flight performance of these vehicle, especially if that control law is based on an experimentally-derived model of that specific UAV configuration. After an accurate model of the flight dynamics is identified and a control law is developed, it must be deployed to the COTS flight hardware of the UAV. This paper demonstrates how to deploy a custom control law to the Pixhawk 4 autopilot on an S500 quad-rotor UAV using the MATLAB & Simulink with the UAV toolbox and PX4 support package, followed by flight test the vehicle in a real world environment.

I. Nomenclature

a_x, a_y, a_z	x, y, z accelerations, m/s/s	T_{rec}	Record length, s
a_0, a_1	Transfer function coefficients	T_{max}	Maximum period, s
g	Acceleration due to gravity, m/s/s	u, v, w	Aircraft x, y, z velocities, m/s
J	Cost function	$\delta_{a,e}$	Aileron, elevator deflection, deg
k	Gain	$\delta_{left,right}$	Elevon deflection, deg
$L_p, L_r, L_v, L_{\delta_{lat}}$	Rolling moment stability parameters	δ_{col}	Throttle input, %
$M_q, M_u, M_w, M_{\delta_{lon}}, M_{\delta_{col}}$	Pitching moment stability parameters	ϕ	Bank angle, deg
N	Scale factor	θ	Pitch angle, deg
$N_p, N_r, N_v, N_{\delta_{lat}}$	Yawing moment stability parameters	τ_c	Time constant, s
$X_q, X_u, X_w, X_{\delta_{lon}}, X_{\delta_{col}}$	X-force stability parameters	τ_δ	Time delay of control input, s
$Y_p, Y_r, Y_v, Y_{\delta_{lat}}$	Y-force stability parameters	ω	Frequency, rad/s
$Z_q, Z_u, Z_w, Z_{\delta_{lon}}, Z_{\delta_{col}}$	Z-force stability parameters	ζ	Damping ratio
p, q, r	Roll, pitch, and yaw rates, deg/s		

*Lieutenant Colonel and Assistant Professor, USAFA-DFAN, 2140 Faculty Drive, USAFA, CO 80840, AIAA member

II. Introduction

ROTORCRAFT and fixed-wing commerical UAV designs have become ubiquitous for entertainment, aerial photography, surveillance, and for classroom instruction and motivation in science, technology, engineering, and mathematics (STEM) fields. Components and modifications are inexpensive and easy to implement thanks to a large online user community.

However, the wide variety of vehicle designs and available modifications can change the flight characteristics of each vehicle. Without tuning or designing an appropriate control law around the actual vehicle configuration, flight performance may be degraded leading to poor station keeping, gust rejection, or battery life. Often the factory-tuned control law is based on the standard vehicle configuration using simple but effective PID gains that don't account for user modifications or mission optimization. Furthermore, the primary goal of commercial manufacturers is to sell a basic flyable aircraft for the widest variety of missions and applications, not to specialize the control law around a particular mission or configuration.

Designing and deploying a custom control law using an experimentally-derived model of the actual flight configuration can result in flight performance benefits such as increased flight duration, longer battery life, improved gust rejection, faster response, and more.

A. Background

Rotorcraft UAVs provide many advantages for a wide variety of mission sets. Primary among those advantages are the ability to take off and land vertically in difficult locations, hover, perform agile maneuvers, and provide a flexible platform for various mission sensors or payloads. Rotorcraft UAVs, which include quad-rotor and multi-rotor aircraft, must be flown with a closed-loop autopilot to stabilize the vehicle dynamics and translate the pilot commands into motor signals that perform the desired maneuver. Popular open-source autopilot software to accomplish this includes ArduPilot and PX4, both of which support many vehicle configurations including multi-copter, fixed-wing, land, and even underwater vehicles. Both provide useful autopilot modes of flight such as position tracking, altitude tracking, and more.

PX4 is an open-source autopilot framework that can control many difference vehicle types and configurations including air, ground, and underwater vehicles. PX4 can integrate a wide variety of sensors and peripheral hardware including companion computers to provide a comprehensive software platform for a wide range of integrated missions, flight modes, and safety features. Together with the QGroundControl ground station software, Pixhawk hardware, and a wide variety of third party hardware components that use the MAVLink protocol, PX4 provides the foundation for a staggering number of vehicles and missions supported by a large group of developers, industry professionals, academics, and hobbyists around the world.

MATLAB Simulink is a block diagram programming environment which allows the user to design, build, and simulate a virtual model of a simple or complex system. The system can be then be integrated onto physical hardware after generating the necessary C or C++ code from the block diagram model in an automated code generation process. Simulink is modular and easily expands to include additional functionality through the use of toolboxes.

B. Theory

The PX4 autopilot can function in a variety of flight modes[1, 2]. The most basic is a manual or stabilized flight mode which provides basic stabilization of the rotorcraft. The pilot's stick inputs from the RC transmitter control the attitude of the vehicle through a PID angle command (for pitch and roll) and a PID rate command (for yaw). The autopilot regulates the pitch and roll angles to zero when the sticks are centered. The throttle setting controls the RPM of the motors, which may hold the desired altitude if the thrust force is balance against the weight and there are no disturbances to the vehicle.

Position and altitude modes are often more useful to the user. In the altitude flight mode, pitch and roll commands are the same as stabilized flight mode but the controller holds the current altitude if the pilot throttle input is around the 50% position. If the throttle input is greater or lower, the controller raises or lowers the altitude of the vehicle. In the position flight mode, centered sticks result in the PID controller holding the current position and altitude of the vehicle.

Designing more advanced control laws requires an accurate dynamic model for accurate simulation, tuning, and testing. Although the control design process is an iterative process, obtaining an accurate and verified model of the aircraft is a critical step to ensure that the control law can be tuned before flight using a simulation which accurately represents the vehicle in flight.

Once a control law is developed and testing in simulation, it can be deployed to the vehicle using the PX4 open-source

flight stack as an easy means to insert the new control laws into an existing and robust framework of sensor estimation and flight routines.

C. Prior Work

This process was accomplished on the 3DR Iris+ airframe in 2015[3]. The author developed a high-fidelity model and used it to the design and implementation of a custom controller designed in Simulink and deployed to the Pixhawk autopilot using the PX4 flight stack. The control law code was auto-generated from MATLAB Simulink and embedded in the Pixhawk firmware to achieve useful flight modes in a safety controlled environment.

Later in 2022, researchers at NASA developed the framework and procedures to achieve flight control law development and rapid testing on small UAVs using the Simulink UAV toolbox and the Pixhawk/PX4 platform with QGroundControl[4]. They demonstrated the deployment and use of a custom attitude stabilization controller integrated into the Pixhawk hardware on a tilt-wing VTOL aircraft with a wingspan around 30 inches.

Significant work was done to obtain an experimentally-derived model of a quadrotor in flight[5]. Flight tests were performed on an S500 quad-rotor aircraft to obtain frequency sweeps in the manual (stabilize) autopilot mode. The controller output was used to identify the bare-airframe vehicle dynamics, both longitudinal and lateral-directional, using system identification in the frequency domain. A valid model was obtained in the frequency range of 1-45 rad/s including time delays representing the motor and system dynamics.

D. Objectives

A streamlined process to develop and deploy custom control laws onto the PX4 flight stack is desired. This process should accommodate an experimentally identified model of the vehicle dynamics so the control law is designed around an accurate model. The final goal is automated code generation from the MATLAB Simulink model into the PX4 flight stack to achieve controlled flight.

III. Methods

A. Flight Vehicle

The S500, depicted in Fig. 1, measures 500mm diagonally from motor to motor and weighs 3 lbs with the battery installed underneath the vehicle. It is powered by 2216-920 kV motors with 1045 propellers and is flown using a Pixhawk 4 autopilot. A GPS module with compass provides position and direction information while a 915 MHz 100 mW radio provides telemetry to the ground station. Piloted flight is achieved with a Jeti hand controller which is capable of switching between flight modes. A flight time of approximately 10 minutes is possible with the 3-cell 5200 mAh LiPo battery.

Fig. 1 S500 Quadrotor UAS

B. Development Environment

The entire development toolchain is contained within MATLAB Simulink with the UAV toolbox, embedded coder, and Simulink coder toolboxes. The UAV toolbox requires an additional support package for PX4 which contains the necessary block diagram functions to interact with the PX4 flight stack[6].

Once the packages are installed, the hardware setup must be completed. This is done through the initial installation of the PX4 support package or accessed through any Simulink diagram through "Hardware Settings". In this step, the Pixhawk autopilot hardware type and version must be selected, as well as the type of build target (PX4 for software-in-the-loop (SITL) or firmware for deployment to the board). The setup process will compile the relevant version of PX4 and upload it to the hardware board, if required. The user can verify the installation by reading the accelerometer values over the USB connection.

C. Procedures

A build-up approach is used to deploy a Simulink model to the Pixhawk 4 autopilot. First, software-in-the-loop (SITL) verification is used in multiple steps to verify the functionality of the Simulink diagram, S500 dynamic model, and closed-loop controller.

The first step is to verify a basic attitude controller designed in Simulink and deployed to the SITL PX4 flight stack. The PX4 flight stack communicates with the built-in jMAVSim to receive user commands through a UDP link with QGroundControl. The SITL-deployed controller sends actuator commands to jMAVSim which contains a basic quad-rotor model that responds and sends motion updates back to the controller. This simulation verifies components of the development environment are working correctly and allows the user to fly the UAV in the jMAVSim visual environment to test the control law design and tuning against a generic quad-rotor model in jMAVSim.

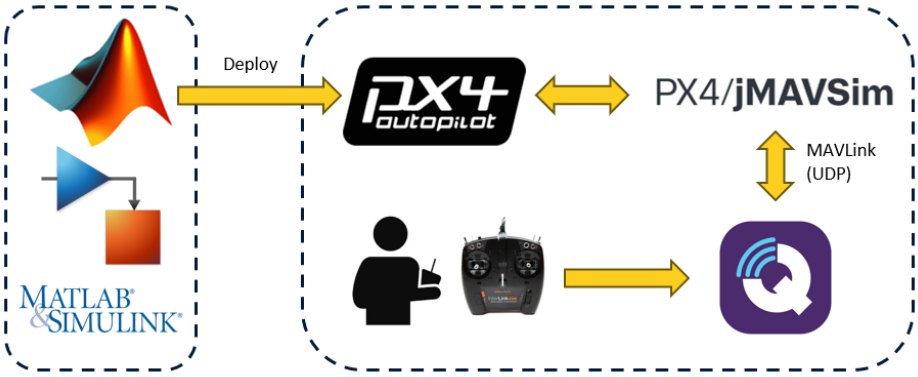


Fig. 2 Software-in-the-loop simulation architecture for the first simulation

The second step builds on the first SITL simulation by adding a position control outer loop and including Simulink in the simulation with the PX4 flight stack, jMAVSim, and QGroundControl. User position commands are input to Simulink which communicates with jMAVSim through the PX4 flight stack. jMAVSim is still connected to QGroundControl so vehicle information and position can be displayed to the user.

The third SITL simulation uses a custom vehicle model in Simulink to provide the vehicle dynamics instead of jMAVSim. This final simulation allows the tuning of the control law around the experimentally-derived flight dynamic model contained in simulink. Two MATLAB instances are required, one to run the Simulink controller and one to run the Simulink vehicle model. User inputs are provided through the controller model while the Simulink model providing the quadcopter dynamics also provides a basic visualization of the motion trajectory.

After verifying the flight controller against the custom vehicle model, it must be deployed to the Pixhawk hardware and verified in a hardware-in-the-loop (HITL) simulation. The architecture is shown in Fig. 5, where the PX4 flight stack with the Simulink tuned controller is deployed to the Pixhawk 4 board. Upon starting the simulation, the Pixhawk board communicates with the custom Simulink flight model over USB, and the Simulink flight model provides motion updates back to the Pixhawk-deployed flight controller based on the vehicle dynamics. The user can input commands or mission coordinates through QGroundControl which sends them to the Simulink model through MAVLink running over UDP.

After the flight controller has been deployed to the PX4 flight stack on the Pixhawk board, the vehicle can be

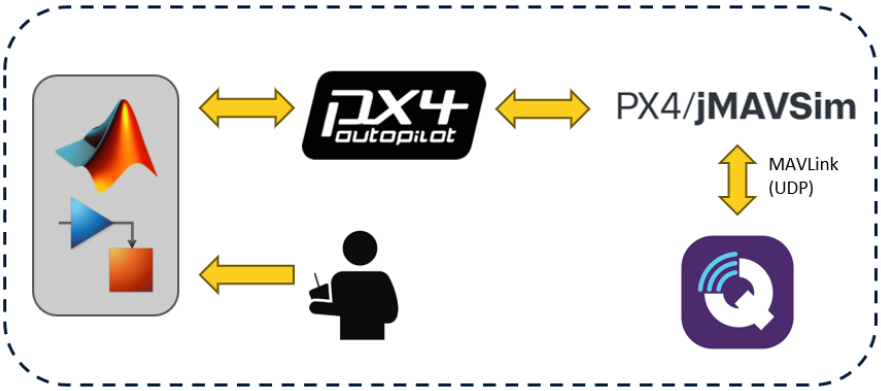


Fig. 3 Software-in-the-loop simulation architecture for the second simulation

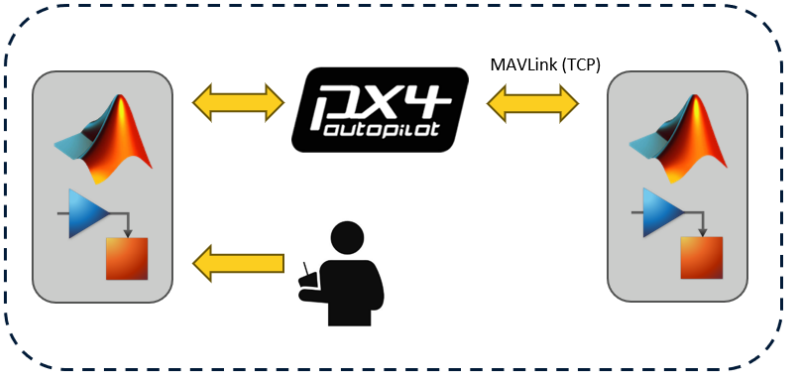


Fig. 4 Software-in-the-loop simulation architecture for the third simulation

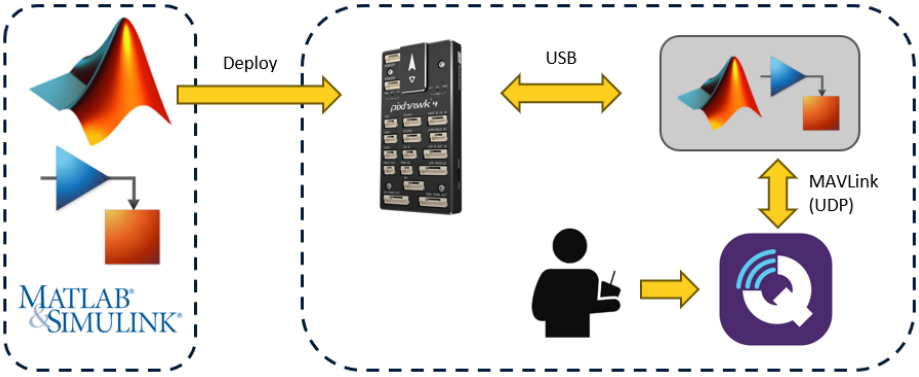


Fig. 5 Hardware-in-the-loop simulation architecture

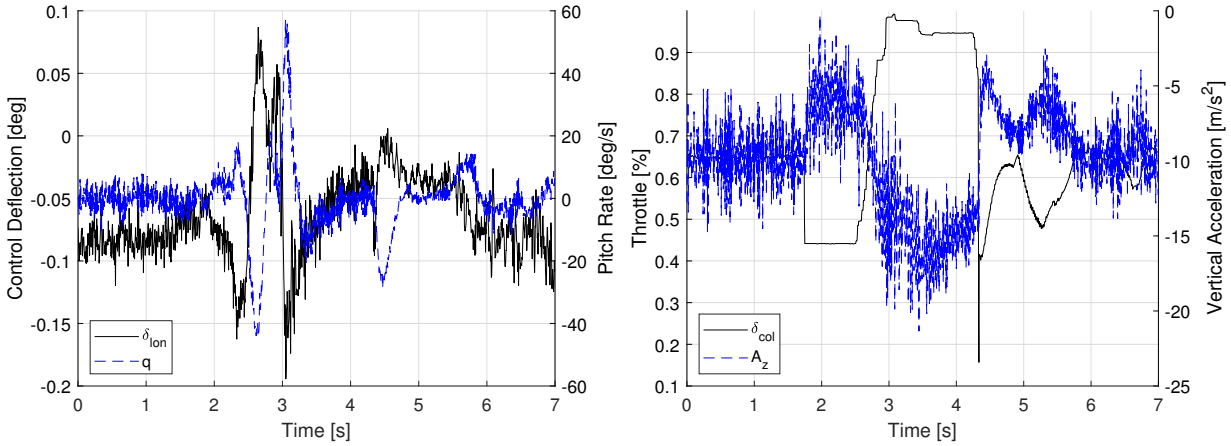
prepared for flight. The hardware configuration must be verified (all Pixhawk connections) as well as the ground station configuration in QGroundControl. Various HITL-specific parameters must be reset in PX4 using QGroundControl, including failsafes and RC transmitter parameters. Finally, the vehicle compass may need to be recalibrated as indicated in QGroundControl.

IV. Results and Discussion

A basic PID controller around attitude and position with custom tuning was used to provide proof-of-concept of the development environment and process. The controller was verified in SITL simulations and then deployed to the vehicle and verified in HITL simulations. After verification, the vehicle was restored to flight-ready condition with the custom-deployed controller.

Flight data obtained from the deployed controller is shown in Fig. 6 and 7. The controller shows good response in each axis, demonstrating the success of the development and deployment process.

In the longitudinal axis, although the signals display some noise, the rise time of the pitch rate response to the control deflection is less than 0.15 seconds. The pitch rate tracks closely to the control deflection throughout the maneuver. In the vertical axis, the accelerometer data display more noise but a nevertheless correct response to the throttle input with a rise time of approximately 0.3 seconds when the commanded throttle is increased.



(a) Pitch rate response to longitudinal input

(b) Vertical acceleration in response to collective input

Fig. 6 Longitudinal aircraft responses

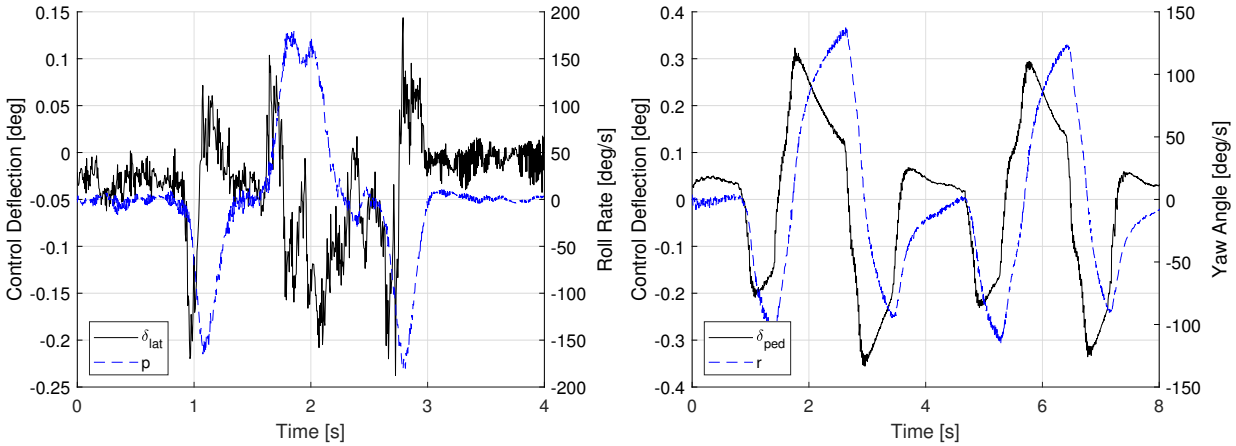
In the lateral axis, the rise time of the response is less than 0.15 seconds and the aircraft tracks closely with the pilot's commands during the maneuver. In yaw, the vehicle and control system have a rise time of less than 0.2 s in yaw angle in response to a pedal input. The yaw angle tracks closely with commanded input through multiple cycles of doublets.

This flight data demonstrates the success of the custom-deployed flight controller running on the Pixhawk 4 autopilot with the PX4 flight stack.

V. Conclusion

Software and hardware in-the-loop simulations were used to design, verify, and deploy a custom flight controller to the Pixhawk 4 autopilot using the PX4 flight stack. The development process was demonstrated with a step-by-step build-up approach which allows the user to verify the desired control law behavior around the actual flight model of the aircraft in question. MATLAB Simulink was used with automatic generation of C++ code along with the build/deploy process of the PX4 firmware. The result is a stable and useful PID controller which showed excellent flight responses in each axis during the flight test.

Future work will implement and test more advanced control laws on the flight vehicle for specific missions and tasks.



(a) Roll rate response to lateral input

(b) Yaw acceleration in response to pedal input

Fig. 7 Lateral aircraft responses

Acknowledgments

The author would like to thank the Edison Grant program of the U.S. Air Force for supporting this work.

References

[1] "PX4 User Guide: Flight Modes," `https://docs.px4.io/main/en/flight_modes/`, 2024. "[Online; retrieved 8-Jan-2024].

[2] "PX4 User Guide: Controller Diagrams," `https://docs.px4.io/main/en/flight_stack/controller_diagrams.html`, 2024. "[Online; retrieved 8-Jan-2024].

[3] Fum, W. Z., "Implementation of Simulink controller design on Iris+ quadrotor," *Naval Postgraduate School*, 2015.

[4] Asper, G. D., and Simmons, B. M., "Rapid Flight Control Law Deployment and Testing Framework for Subscale VTOL Aircraft," *National Aeronautics and Space Administration*, 2022.

[5] Babcock, J. T., "System Identification of an S500 Quadrotor UAV," *Defense Technical Information Center*, 2023.

[6] "UAV Toolbox Support Package for PX4 Autopilots," `https://www.mathworks.com/help/supportpkg/px4/`, 2024. "[Online; retrieved 8-Jan-2024].