Full-Stack UAV Simulator for the Cessna 172

Adin Sacho-Tanzer and Gabriel Kret
adin.sachotanzer@cooper.edu
gabriel.kret@cooper.edu
ME-457 Drone Control
Instructor: Dr. Luchtenberg

May 16, 2025

Contents

1 Introduction

Our objective was to simulate the flight dynamics and control of a fixed-wing aircraft, initially the Aerosonde UAV, then reconfigured for a general aviation aircraft, the Cessna 172. The final product includes realistic modeling of aerodynamics, a custom engine model, full PID control via Successive Loop Closure, simulated sensors with Gaussian noise, and a two-stage Extended Kalman Filter for state estimation.

Initially, the Aerosonde UAV was modeled using 6-degree-of-freedom equations of motion in quaternion form, with aerodynamic models and basic control functionality implemented. We then extended this baseline system to include realistic sensor modeling, a full autopilot architecture based on successive loop closure (SLC), and a Kalman Filter for state estimation. After this was done, we transitioned to modeling the general aviation aircraft Cessna 172. This required sourcing new physical and aerodynamic parameters, implementing a different engine model, recalculating trim and linearization conditions, and retuning both controllers and estimators to accommodate the new flight dynamics.

This memo details each component involved in the simulation. This includes the rigid body dynamics, the aerodynamics, the controller, and the Kalman filter. The final deliverable was a fully autonomous flight sequence, from takeoff to landing, executed within the simulator using only sensor-derived state estimation, robust control loops, and accurate modeling of wind, propulsion, and aerodynamic interactions.

2 Architecture

The simulator was developed in Python and is based on the `mavsim_public` codebase from Beard and McLain [1], which provides a modular and extensible environment for modeling UAV dynamics, control systems, and state estimation. We made additions and modifications to adapt the simulator for different functionality and the transition to a Cessna 172 airframe.

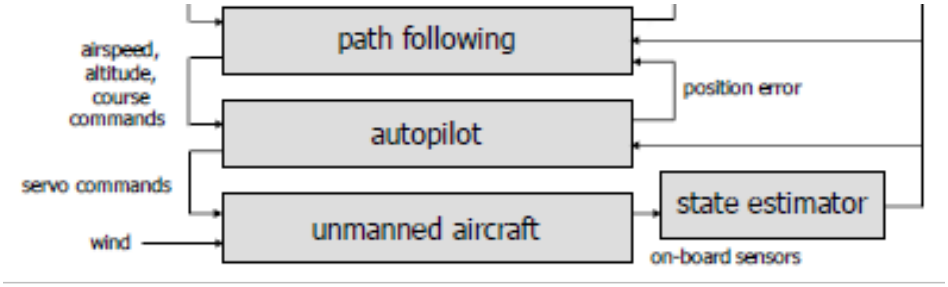


Figure 1: Software Architecture Flowchart [4]

The architecture is built around object-oriented principles. Each subsystem (dynamics, sensors, estimation, control, and plotting) is encapsulated in its own module. Some of the key files that make up these subsystems include:

- **aerosonde_parameters.py**: Contain physical constants and tuning parameters, customized for the Aerosonde (or Cessna 172). The parameters for the Aereosonde are found in *Small Unmanned Aircraft: Theory and Practice* [4]. The Cessna 172 Parameters were found in *Cessna 172 Flight Simulation Data* [5] and *Investigation of Multi-Input Multi-Output Robust Control Methods...* [2]. See Appendix B for the parameters used for the Cessna 172 Model.
- **control_parameters.py**: Contains tuning parameters for the PID controller.
- **mav_dynamics.py**: Contains the rigid-body equations of motion (EOMs), using quaternion state representation to avoid singularities.
- **mav_dynamics_control.py**: Extends the base dynamics to compute aerodynamic and propulsion forces/moments.
- **mav_dynamics_sensors.py**: Adds realistic sensor emulation, including bias, noise, and sampling

rates.

- **autopilot.py**: Implements longitudinal and lateral PID controllers using a Successive Loop Closure (SLC) architecture.
- **observer.py**: Houses the EKF estimator that combines the simulated sensor data and mathematical model into a state estimate.
- **takeoff_sim.py**: An autonomous mission from takeoff to landing using logic-driven transitions.

By the original authors design, modules are decoupled by message-passing classes, mimicking a real-world onboard flight software structure. This allowed us to integrate components like the observer and autopilot independently and test them in isolation [1].

3 Dynamics

The UAV dynamics are modeled using the full 6-DOF rigid-body equations of motion derived in Beard and McLain's UAV textbook. The state vector includes translational and angular velocities, position, and orientation expressed in quaternion form. The Equations of Motion can be found in Appendix A.1. The equations of motion are integrated using a fourth-order Runge-Kutta (RK4) scheme for numerical accuracy. Forces and moments acting on the airframe are calculated in the body frame and transformed to the inertial frame as needed. Code exists to simulate wind effects as both steady-state and gust components, however for the following testing, wind was uniformly set to 0.

4 Aerodynamics

Aerodynamic forces and moments were computed using a linear aerodynamic model. These models calculate lift, drag, and side forces based on angle of attack, sideslip, and control surface deflections. The general force equations can be found in Appendix A.2.

5 Trim

Trim conditions were determined using a nonlinear optimization process implemented in **'run_trim_sim.py'**. The goal was to find a state and a set of control positions where the aircraft maintains steady-level flight, i.e., all state derivatives except for position are zero or within acceptable tolerance. For the Cessna, trim was calculated for a target airspeed of 62.8 m/s, a typical cruising speed for this aircraft.

When running the Aerosonde model, a persistent small nonzero pitch rate ($\dot{q}$) was observed at the trim condition. This was attributed to numerical inaccuracy rather than physical instability. The resulting trim point was used to linearize the dynamics via Jacobian approximations, forming the basis for control design and Kalman filtering. The trim state computed for the Cessna 172 did not suffer from this inaccuracy and can be found in Section 9.2.

6 Controller

The control system is based on Successive Loop Closure (SLC), where inner-loop dynamics are stabilized before tuning outer loops. If the frequency at which the outer loops operate is much greater ($\sim 20x$) than that of the inner loops, the inner loop can be considered "closed", because its control operates on a much faster timescale than the outer loop. When this happens, the inner loop effectively serves as a DC gain on the outer loop, allowing for easier control. We implemented:

- **Roll:** PD controller using aileron.
- **Pitch:** PD controller using elevator.
- **Yaw:** Transfer function-based yaw damper using rudder.
- **Course:** PI controller using roll command.
- **Altitude:** PI controller using pitch command.
- **Airspeed:** PI controller using throttle.

To calculate gains for each of these controllers, the linearized system was analyzed using transfer functions. This allowed us to derive formulas for the frequencies (ω) and damping coefficients (ζ) of each system in

terms of the gains. With these calculations, it is possible to calculate the PID gains of a system in terms of ζ and ω, allowing these parameters to be tuned instead of tuning the gains directly. Final tuning achieved stable and responsive tracking in all loops.

7 Sensors

Sensor simulation included gyros, accelerometers, magnetometers, GPS, and pressure sensors. Each sensor was augmented with additive Gaussian noise and realistic sampling rates. These sensors enabled realistic estimation testing, and required robust filtering using low-pass and kalman filters to suppress noise and maintaining accuracy.

8 Kalman Filter

Two Extended Kalman Filters (EKFs) were implemented: one for estimating attitude (phi, theta) and another for position and navigation states (pn, pe, chi, Vg, wn, we). The filters used continuous-discrete formulations and handled delayed GPS updates via conditional updates.

A major bug was found in the original EKF implementation by Beard involving misordered inputs. After correction, the EKF performance significantly improved, exhibiting accurate convergence under sensor noise.

Plots showing our controller before and after applying the Kalman Filter can be found in Appendix C.

9 Transition to the Cessna 172

For the final project, we transitioned from the Aerosonde UAV to the Cessna 172. This required multiple system-level changes:

- **Aerodynamic parameters:** Updated to Cessna 172 using values from *Cessna 172 Flight Simulation Data* [5] Stability Derivatives were collected from *Investigation of Multi-Input Multi-Output Robust Control Methods* [2]. The Stability Derivatives and Aircraft Properties can be found in Appendix B.
- **Engine model:** Replaced DC electric motor with a propeller thrust model based on engine power output.
- **Re-tuning:** PID and Kalman filter parameters adjusted for new inertia and aerodynamic properties.

The engine model we used modeled thrust as follows:

$$T = \frac{P_{engine} \cdot \eta \cdot (A_p \frac{\rho}{\rho_{SL}} - B_p)}{V_a}$$

where:
- T is thrust produced by the engine
- P_{engine} is the power produced by the engine
- η is the efficiency of the engine
- A_p and B_p are constants, equal to 1.132 and 0.132, respectively
- ρ and ρ_{SL} are the atmospheric density at the aircraft's position and at sea level, respectively
- V_a is the airspeed of the aircraft

In order to implement this model, the following assumptions were made:

- $P_{engine} = \max(\delta_T \times 134kW, 0.05 \times 134kW)$. The maximum power output of the Cessna 172 engine is 134 kW [9]. This model assumes that our throttle linearly varies the power output of our engine with a minimum value of 5%. This minimum is in order to prevent divide-by-zero errors, and is realistic because the engine would not be turned off in flight in real life.
- η was taken to be 0.8, which is a typical value for propellor engine efficiency [10].

- $\rho_{SL} = \rho$. This assumes that the air density at the aircraft's altitude is the same as the air density at sea level. This assumption was made because throughout the rest of the aircraft dynamics, ρ was assumed constant, so this assumption was consistent with that.
- We further assumed that the engine applies only linear thrust, and not torque, to the aircraft. While this assumption is a somewhat crude approximation, the torque is small compared to the thrust, so it was assumed to be negligible.

Additionally, the derivatives $\frac{\partial T}{\partial V_a}$ and $\frac{\partial T}{\partial \delta_T}$ were needed in order to form a transfer function model of the aircraft dynamics, necessary for tuning our PID controller in terms of ζ and ω instead of PID gains. These were computed according to our engine model as:

$$\frac{\partial T}{\partial V_a} = (-134kW)\delta_T(A_p - B_p)\frac{\eta_p}{V_a^2}$$

$$\frac{\partial T}{\partial \delta_T} = 134kW(A_p - B_p)\frac{\eta_p}{V_a}$$

9.1 Test Scenario

The scenario used for final validation of the Cessna consisted of:

1. Accelerated takeoff at 28 m/s.
2. Climb to 100 m.
3. Level cruise at 62.8 m/s.
4. Coordinated right turn (90° heading).
5. Descent and simulated landing.

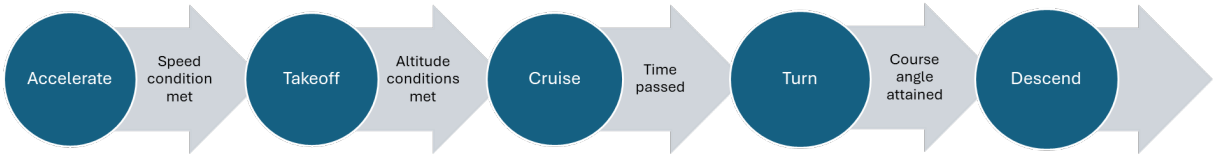


Figure 2: Flight mode transitions during autonomous mission scenario.

A simple ground model was implemented to prevent the aircraft from "falling" through the terrain:

```
if mav.true_state.altitude <= 0.0:
    mav.true_state.altitude = 0.01
    mav._state[2] = 0.01
```

While crude, it was sufficient for flight profile testing.

Logic transitions were encoded as states in 'takeoff_sim.py'. Each phase of flight was executed with full estimation and control, without access to ground-truth state data.

Dynamic PID Gain Scheduling

To improve aircraft response across varying flight phases, we implemented *dynamic PID tuning*, where controller gains are adjusted in real time based on the mission stage. This allows each segment of the flight

profile to be optimized independently, accommodating the different dynamic requirements of phases such as takeoff, climb, cruise, turning, and landing.

For example:

- During the **Cruise** phase, a moderate course damping ratio of `CP.zeta_course = 0.6` was used to ensure stable tracking without aggressive corrections.
- Upon entering the **Turn** phase, we increase the zeta, and by extension the proportional gain, on the course loop, leading to better and faster tracking of the course command. This k_p was greatly increased to achieve rapid heading changes with minimal overshoot. The results of this are shown in Figure 3.
- Similarly, upon exiting the turn and going into **Landing**, `CP.zeta_course` was dropped back to its initial value and `CP.zeta_pitch` was increased to tighten vertical control for precise altitude tracking as shown in Figures 3 and 7.

Figure 3: Effect of dynamic PID tuning on course response. The transition to a higher damping ratio during the "Turn" phase improves response time and reduces overshoot. As you can see, when we drop the damping coming out of the turn, this leads to some small oscillations. This figure will be referenced further in Section 9.2.

This mission-aware gain scheduling resulted in improved responsiveness where needed, such as sharper course convergence during turns, while preserving stability and smooth transitions during other phases like climb and cruise. This is a good technique to use when designing a common procedure used by an autopilot, such as takeoff and landing. By adapting gains on-the-fly, we demonstrated that a single PID architecture can be extended for complex flight scenarios without introducing instability or requiring multiple separate controllers.

It is worthwhile to note that the model does not include airbrakes, flaps, or other mechanisms designed to decelerate the aicraft, however, by extending the length of the landing, we were able to slowly reduce our speed to 30m/s which is only slightly faster than the recommended landing speed of a Cessna 172. The result was fully autonomous mission execution.

9.2 Results

All systems performed as expected. Highlights:

Trim Validation: All relevant state derivatives were close to zero, confirming proper trim convergence.

Trim State	State Derivative	Control Inputs
$\begin{bmatrix} 2.2148 \times 10^{-16} \\ 1.3860 \times 10^{-15} \\ -4.5548 \times 10^{-15} \\ 6.2796 \times 10^{1} \\ 0.0000 \\ -6.6730 \times 10^{-1} \\ 9.9999 \times 10^{-1} \\ 0.0000 \\ -5.3130 \times 10^{-3} \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix}$	$\begin{bmatrix} 6.2800 \times 10^{1} \\ 0.0000 \\ 9.6709 \times 10^{-7} \\ 2.0771 \times 10^{-1} \\ -1.3035 \times 10^{-7} \\ -1.0893 \times 10^{0} \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 4.6371 \times 10^{-7} \\ -2.8561 \times 10^{-1} \\ 2.7505 \times 10^{-7} \end{bmatrix}$	$\begin{aligned} \text{elevator} &= -0.00433 \\ \text{aileron} &= -9.9431 \times 10^{-9} \\ \text{rudder} &= -1.9707 \times 10^{-8} \\ \text{throttle} &= 0.69532 \end{aligned}$

Trim State, Trim State Derivatives, and Control Inputs for the Cessna 172 Model.

PID Control: Achieved steady-state tracking with minimal overshoot and acceptable rise time.

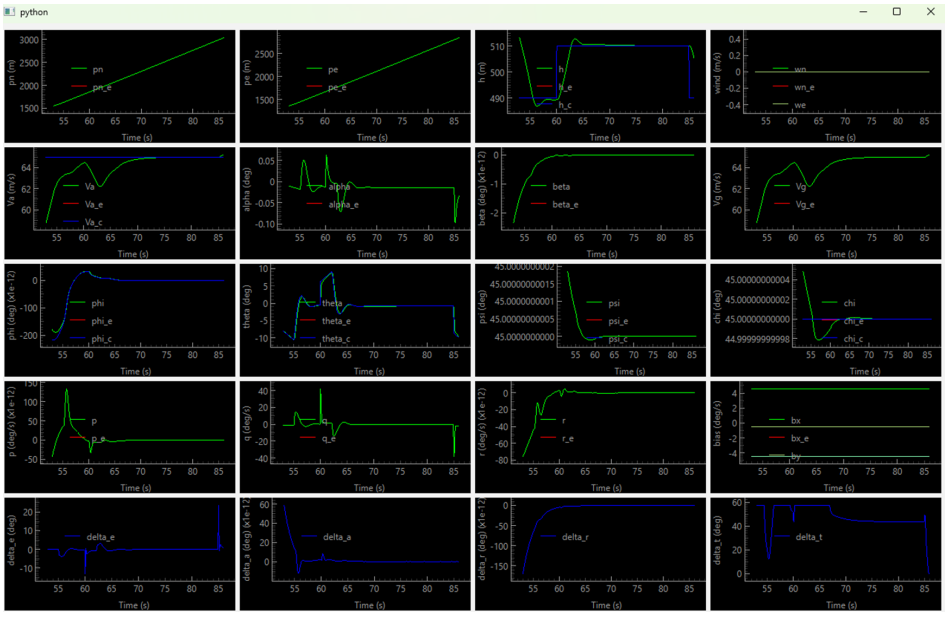


Figure 4: PID performance: altitude, airspeed, and course over time. The blue line represents the commanded properties while the green represents the true properties. The plots show how, for each commanded property, the true properties converge to the command with minimal overshoot and rise time.

Kalman Filter: After making the corrections indicating in Section 8, EKF convergence was rapid and stable under noise and GPS delay.

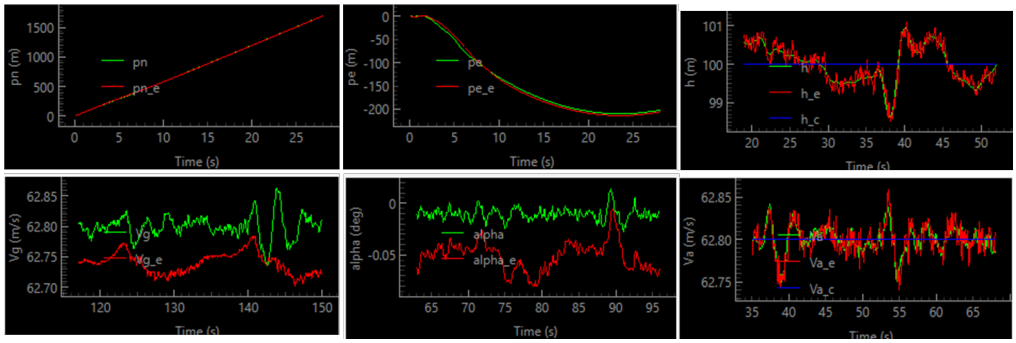


Figure 5: Kalman filter convergence for selected position and attitude states. Additional plots showing the Kalman Filters response can be found in Figure 10 of Appendix C.

Autonomous Flight: After all systems were implemented the full autopilot was tested and examined for instability. The plane remained stable and proceeded through its objectives safely.

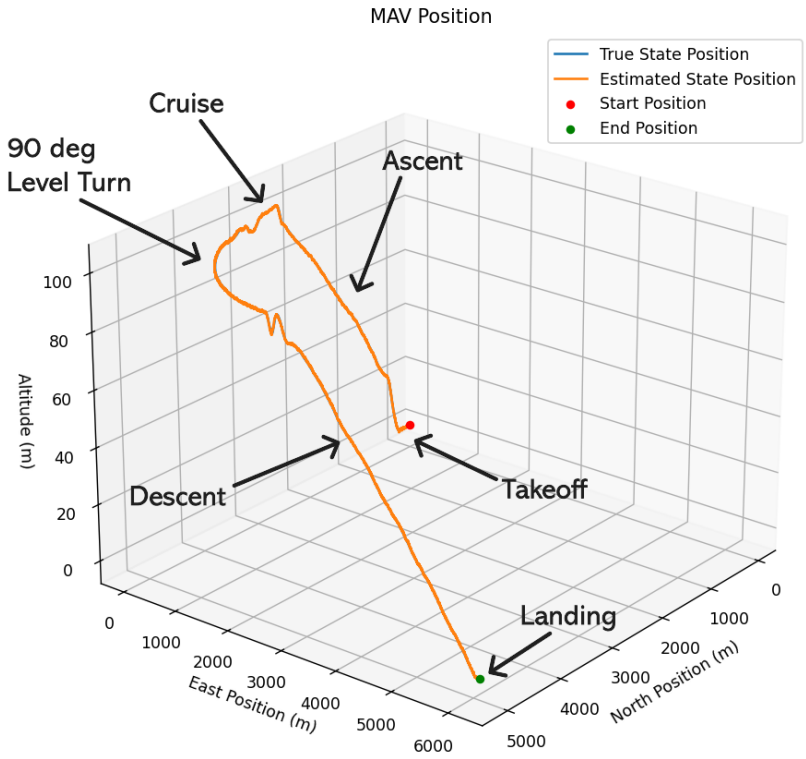


Figure 6: Sample flight trajectory from takeoff to landing.

Flight profiles in Figure 6 confirmed the full system's integrity and accuracy under realistic simulation conditions. Figure 7 shows that the controller still tracks altitude well, even with the added sensor noise. Finally, Figure 3, shown earlier in Section 9.1 as part of the Dynamic PID Control implementation, shows how the course angle is tracked well with the Kalman Filter Implemented. It is notable that in Figure 3, the best tracking occurs while the course angle is being adjusted and oscillation begin to occur once the course command stabilizes. This is because of the dynamic PID adjustment, which increases k_p when an adjustment is being made, once we stop adjusting the course angle, we drop the proportional gain, this leads to increased oscillation which we take as a trade off for other parameters deemed more important for steady flight in that mission stage.

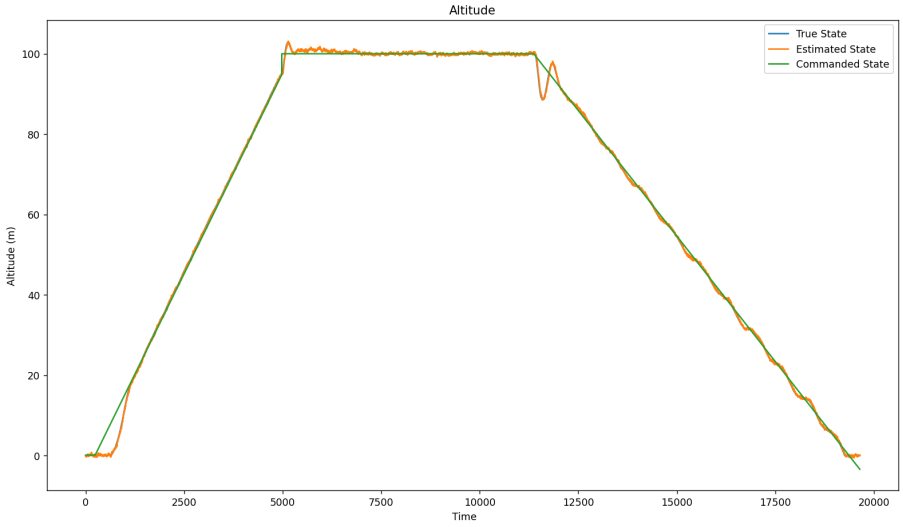


Figure 7: Autonomous flight profile: altitude vs. time showing takeoff, climb, cruise, descent. The curve for the True State is hidden beneath the estimated state meaning our Kalman Filter tracks the True State well.

9.3 Challenges

Key difficulties in our Cessna 172 model development included:

- Incorrect filter implementation (solved by code audit).
- Lack of empirical data for Cessna propulsion (approximated with Lutze model [3]).
- PID and Kalman Filter tuning is tedious and extremely sensitive to dynamics and interactions.

These challenges required time-consuming debugging and iterative redesign.

10 Conclusion

We developed a complete drone simulator capable of simulating, estimating, and controlling a fixed-wing aircraft through a full mission profile. The transition from Aerosonde to Cessna 172 added realism and complexity, handled through careful design and a lot of brute-force debugging.

This project is an example of the intersection of dynamics, control, estimation, and software engineering. The final simulator is flexible and demonstrates high quality simulations in flight control scenarios.

11 Future Work

Recommendations for improvement:

- Add ground interaction models (friction, tipping).
- Introduce flap and airbrake controls.
- Support stall modeling and nonlinear aerodynamics.
- Implement auto-tuning algorithms.

These modifications would increase realism and make the simulator more accurate so that we can model more advanced flight scenarios.

Appendices

A Equations

A.1 Rigid Body Dynamics

Here are the EOM's in quaternion format used for the systems dynamics (See Section 3):

$$\begin{pmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{p}_d\end{pmatrix} = \begin{pmatrix}e_1^2+e_0^2-e_2^2-e_3^2 & 2(e_1e_2-e_3e_0) & 2(e_1e_3+e_2e_0)\\ 2(e_1e_2+e_3e_0) & e_2^2+e_0^2-e_1^2-e_3^2 & 2(e_2e_3-e_1e_0)\\ 2(e_1e_3-e_2e_0) & 2(e_2e_3+e_1e_0) & e_3^2+e_0^2-e_1^2-e_2^2\end{pmatrix}\begin{pmatrix}u\\ v\\ w\end{pmatrix}$$

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

$$\begin{pmatrix}\dot{e}_0\\ \dot{e}_1\\ \dot{e}_2\\ \dot{e}_3\end{pmatrix} = \frac{1}{2}\begin{pmatrix}0 & -p & -q & -r\\ p & 0 & r & -q\\ q & -r & 0 & p\\ r & q & -p & 0\end{pmatrix}\begin{pmatrix}e_0\\ e_1\\ e_2\\ e_3\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix} = \begin{pmatrix}\Gamma_1pq-\Gamma_2qr\\ \Gamma_5pr-\Gamma_6(p^2-r^2)\\ \Gamma_7pq-\Gamma_1qr\end{pmatrix} + \begin{pmatrix}\Gamma_3l+\Gamma_4n\\ \frac{1}{J_y}m\\ \Gamma_4l+\Gamma_8n\end{pmatrix}$$

where:

$$\Gamma = J_xJ_z - J_{xz}^2,\quad \Gamma_1 = \frac{J_{xz}(J_x-J_y+J_z)}{\Gamma},\quad \Gamma_2 = \frac{J_z(J_x-J_y)+J_{xz}^2}{\Gamma},$$
$$\Gamma_3 = \frac{J_z}{\Gamma},\quad \Gamma_4 = \frac{J_{xz}}{\Gamma},\quad \Gamma_5 = \frac{J_z-J_x}{J_y},\quad \Gamma_6 = \frac{J_{xz}}{J_y},$$
$$\Gamma_7 = \frac{(J_x-J_y)J_x+J_{xz}^2}{\Gamma},\quad \Gamma_8 = \frac{J_x}{\Gamma}$$

A.2 Aerodynamic Forces and Moments

Here are the equations for the Aerodynamic forces on the Aircraft (See Section 4):

$$F_{\text{lift}} = \frac{1}{2}\rho V_a^2 S\left[C_{L_0} + C_{L_\alpha}\alpha + C_{L_q}\frac{c}{2V_a}q + C_{L_{\delta_e}}\delta_e\right]$$
$$F_{\text{drag}} = \frac{1}{2}\rho V_a^2 S\left[C_{D_0} + C_{D_\alpha}\alpha + C_{D_q}\frac{c}{2V_a}q + C_{D_{\delta_e}}\delta_e\right]$$
$$m = \frac{1}{2}\rho V_a^2 Sc\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e\right]$$
$$f_y \approx \frac{1}{2}\rho V_a^2 S\left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\right]$$
$$l \approx \frac{1}{2}\rho V_a^2 Sb\left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r\right]$$
$$n \approx \frac{1}{2}\rho V_a^2 Sb\left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right]$$

B Cessna Parameters

Cessna 172 Physical and Aerodynamic Parameters [2][5]

Parameter	Value	Units
Mass (m)	1043.3	kg
Moment of inertia about x-axis (J_x)	1285.3	kg·m^2
Moment of inertia about y-axis (J_y)	1824.9	kg·m^2
Moment of inertia about z-axis (J_z)	2666.9	kg·m^2
Product of inertia (J_{xz})	0.0	kg·m^2
Wing area (S)	16.1651	m^2
Wingspan (b)	10.9118	m
Mean aerodynamic chord (c)	1.4935	m
Air density (ρ)	1.2682	kg/m^3
Aspect ratio (AR $= b^2/S$)	7.37	–
Gravitational acceleration (g)	9.81	m/s^2
Maximum engine power ($P_{\max}$) [6]	134,000	W
Propeller efficiency (η) [7]	0.8	–
Propeller coefficient A_p [8]	1.132	–
Propeller coefficient B_p[8]	0.132	–

Specified Cessna 172 Stability Derivatives for the Aerodynamic Model [2]

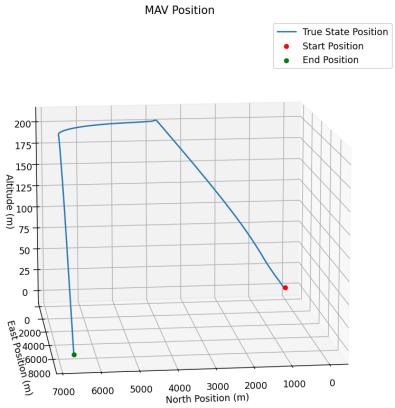
Longitudinal Derivatives

Derivative	Value
C_{D_0}	0.031
C_{D_α}	0.13
C_{D_q}	0
$C_{D_{\delta_e}}$	0.06
$C_{D_{\delta_h}}$	0
C_{L_0}	0.31
C_{L_α}	5.143
C_{L_q}	3.9
$C_{L_{\delta_e}}$	0.43
$C_{L_{\delta_h}}$	0
C_{m_0}	-0.015
C_{m_α}	-0.89
C_{m_q}	-12.4
$C_{m_{\delta_e}}$	-1.28
$C_{m_{\delta_h}}$	0

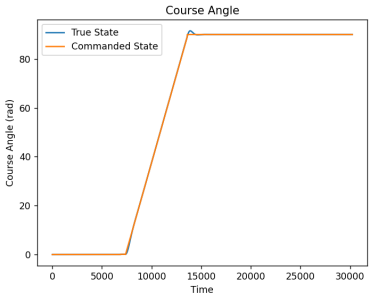
Lateral-Directional Derivatives

Derivative	Value
C_{Y_β}	-0.31
C_{Y_p}	-0.037
C_{Y_r}	0.21
$C_{Y_{\delta_a}}$	0.0
$C_{Y_{\delta_r}}$	0.187
C_{l_0}	0
C_{l_β}	-0.089
C_{l_p}	-0.47
C_{l_r}	0.096
$C_{l_{\delta_a}}$	-0.178
$C_{l_{\delta_r}}$	0.0147
C_{n_0}	0
C_{n_β}	0.065
C_{n_p}	-0.03
C_{n_r}	-0.099
$C_{n_{\delta_a}}$	-0.053
$C_{n_{\delta_r}}$	-0.0657

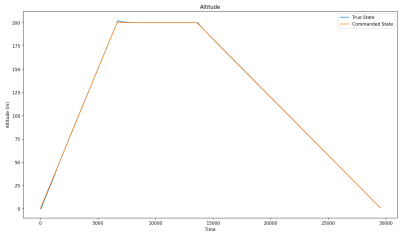
C Additional Figures



(a) PID Only: Aircraft Position



(b) PID Only: Course Angle Tracking before Kalman Filter



(c) PID Only: Altitude Tracking before Kalman Filter

Figure 8: Plots showing aircraft control before a Kalman Filter is applied shows that our PID Controller is well tuned. Analogous plot with the Kalman Filter applied are shown throughout the report in Figures 3, 6, and 7.

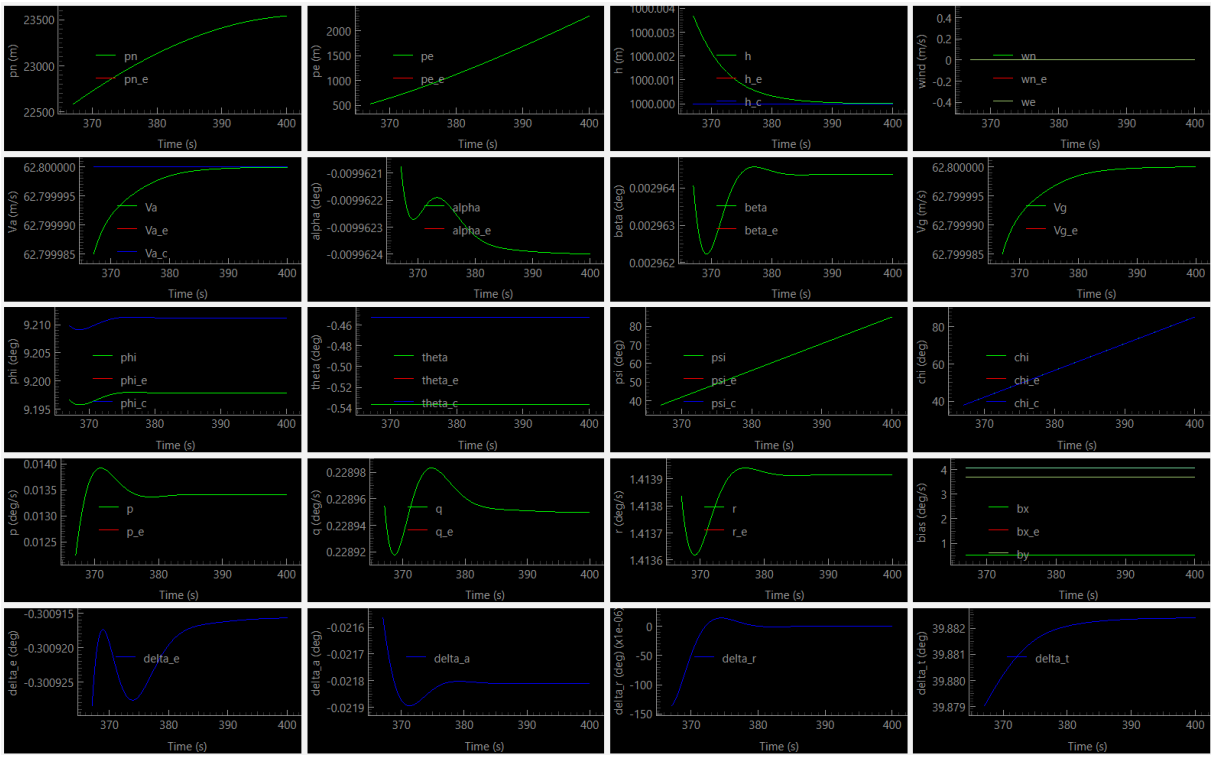


Figure 9: PID Controller Response without Kalman Filter on Takeoff-Landing Test Case.

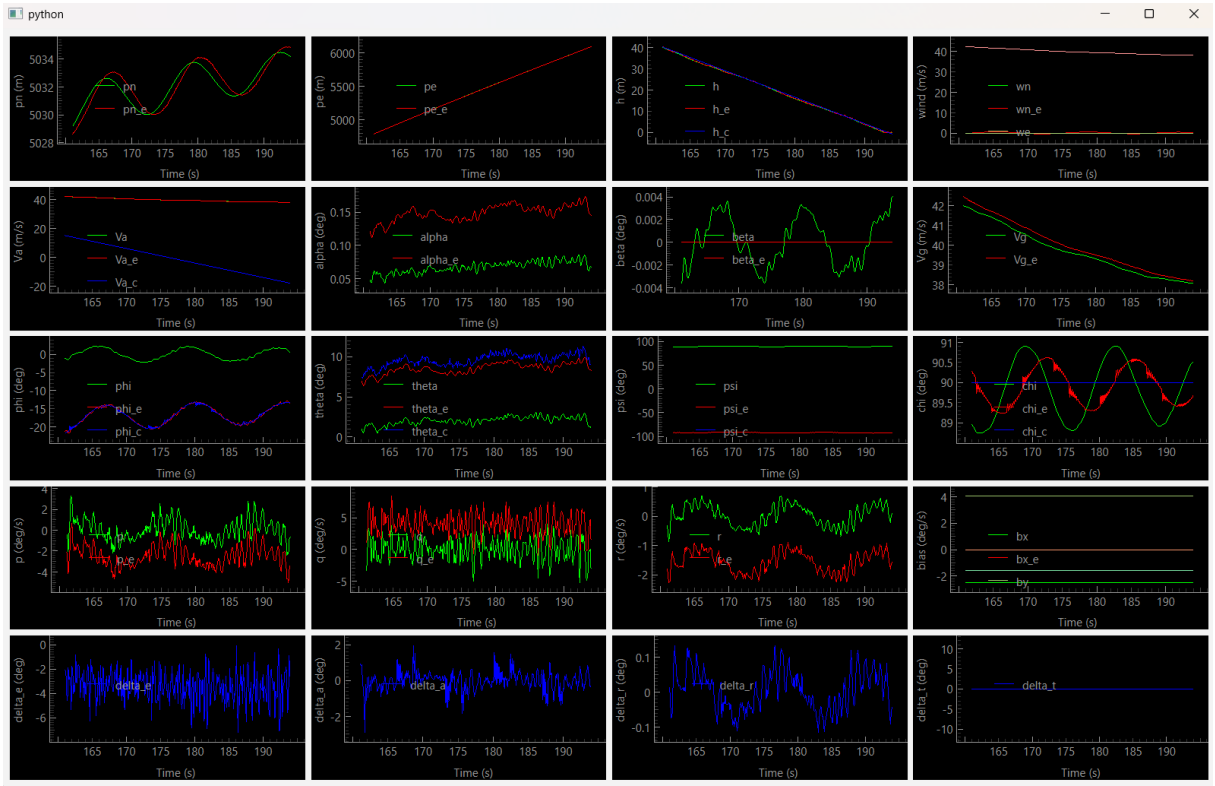


Figure 10: Kalman Filter estimate convergence over time.

D Supplementary Video

A full demonstration video of the simulated flight scenario is available at:

`https://youtu.be/aoBvvNAL3Po`

E GitHub Repository

The complete source code, simulation parameters, and project documentation are publicly available in the project's GitHub repository at the following URL:

`https://github.com/gkret123/ME457-Drone-Control`

The key repository sections include:

- `Kret_Sacho-Tanzer_Simulator_Quaternion`: Contains the Aerosonde drone simulator developed over the course of the semester, including quaternion-based equations of motion, PID control, and Kalman filter implementation.
- `Final_Project_Kret_Sacho-Tanzer_Simulator_Quaternion`: Includes the adapted simulator specifically modified for the Cessna 172 aircraft, featuring updated aerodynamic parameters, propulsion system modeling for a gasoline engine, recalculated trim conditions, retuned PID controller, and refined Kalman filter settings.
- `Documentation`: Houses all relevant project documentation including technical reports, presentations, diagrams, and additional references essential for understanding and replicating the project.

F References

[1] R. Beard, *mavsim_public*, GitHub Repository: https://github.com/byu-magicc/mavsim_public.

[2] C. Kasnakoglu and I. Cosku, "Investigation of Multi-Input Multi-Output Robust Control Methods," PLOS ONE, 2016.

[3] F. H. Lutze, *Thrust Models*, Virginia Tech Dept. of Aerospace Engineering. https://archive.aoe.vt.edu/lutze/AOE3104/thrustmodels.pdf

[4] R. Beard and T. McLain, *Small Unmanned Aircraft: Theory and Practice*, 2nd ed., Princeton University Press, 2024.

[5] M. M. Cel, *Cessna 172 Flight Simulation Data*, Revision 9, 2019. Available at: https://www.researchgate.net/publication/353752543_Cessna_172_Flight_Simulation_Data

[6] Wikipedia contributors. *Cessna 172*. Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Cessna_172

[7] Google Search. *Typical propeller efficiency values*. Retrieved from general aviation resources.

[8] Lutze, F. H. *Thrust Models*. Virginia Tech. Retrieved from https://archive.aoe.vt.edu/lutze/AOE3104/thrustmodels.pdf

[9] Wikipedia contributors,"Cessna 172," *Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/wiki/Cessna_172, accessed May 15, 2025

[10] MIT Unified Engineering, "Control Volume Analysis," *Unified Engineering Thermodynamics Notes*, https://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node86.html, accessed May 15, 2025.