https://arc.aiaa.org/doi/abs/10.2514/6.2020-1238

Quadrotor Drone System Identification via Model-Based Design and In-Flight Sine Wave Injections

Dr. Daniel S. Kaputa* and Keith J. Owens†

Precise dynamic system characterization of a small indoor drone is performed using a camera-based motion tracking system. Model-based design is used to target a multi-loop controller to a drone directly from a Simulink model via External Mode, and to correlate the model with the hardware using discrete frequency sinewave inputs in all axes. Finally, autonomous flight is achieved using the camera-derived position and attitude as feedbacks.

I. Nomenclature

K_t	=	Motor torque coefficient, N m/amp
K_e	=	Motor back emf coefficient, V/(rad/s)
V_{batt}	=	Battery voltage, volt
R_{tt}	=	Motor resistance (terminal-terminal), ohm
J_m	=	Motor and prop inertia, kg m²
D_r	=	Rotor (prop) diameter, m
ρ	=	Air density, kg/m³
T	=	Propeller thrust, N
Q	=	Propeller torque, N m
C_T	=	Propeller thrust constant
C_P	=	Propeller power constant
Ixx, Iyy, Izz	=	Drone moments of inertia, kg m²
m	=	Drone mass, kg
L_x, L_y	=	Moment arms from CG to motors, m
$\omega_x, \omega_y, \omega_z$	=	Body axis rotation rates, rad/s
ψ, θ, ϕ	=	Inertial-to-body axis Euler angles, rad
u_x, u_y, u_z	=	Body axis velocities at sensed location
$u_{x_{cg}}, u_{y_{cg}}, u_{z_{cg}}$	=	Body axis velocities at CG
ω_m	=	Motor speed, rad/s
T_d	=	Hardware update delay, inertial measurement unit (IMU)
T_{d2}	=	Hardware update delay, OptiTrack feedbacks
CG	=	Center of gravity
z_{cg}	=	Vertical center of gravity distance beneath OptiTrack sensor measurement point
G^{output}_{input}	=	Transfer function from input to output

II. Introduction

Most commercial quadrotor UAVs come with flight controllers pre-tuned for a specific weight or payload. When one wishes to build a new UAV or design a custom payload for a UAV similar to that developed by Kaputa Et al. [1] the canned PID gains of the flight controller may be suboptimal and result in instability, reduced flight time, or even crashing. Off-the-shelf flight controllers such as Ardupilot offer two main ways to tune the system gains, namely manual tuning and autotuning [2]. The problem with these methods is that manual tuning rarely arrives at the optimal gains and both methods have an inherent risk of inducing instabilities which could result in a crash.

*Assistant Professor, Rochester Institute of Technology, Electrical, Computer, and Telecommunications Engineering Technology, 78 Lomb Memorial Drive, Rochester, NY, 14623.

†Staff Systems Engineer, Moog Space and Defense Group, Technology and Advanced Pursuits, 500 Jamison Rd, East Aurora, NY 14052.

The process proposed in this article is different than both manual and autotuning approaches in that the goal is to use information generated from injected sine waves to perform system identification in order to determine the exact physics of the UAV. A logical outcome of having an accurate high fidelity system model is that one can better determine what the appropriate gains should be. Not only can optimal gains be generated but having a grasp of the dynamic system model can give the developer insights into questions such as "How would increasing the prop size affect the system stability?" Lastly, as shown in Fig. 1, once the UAV dynamic model is completely understood, custom flight controllers can be created for specific applications.

Although there have been great advances in the area of model-free control [3] [4], traditional methods of establishing closed-loop control over an inherently unstable air vehicle require a thorough understanding of the dynamics of the system. A flight dynamics model is typically derived from well-known equations of motion, however there can be large variations in model parameters such as aerodynamic coefficients, inertia, mass, etc. To get a good estimate of these parameters based on the topology of the air vehicle under study, many researchers use models that are contained within programs such as X-Plane (NASA) and Flight Simulator X (Microsoft) [5]. Another way to determine model parameters is to perform experiments ranging from single-motor thrust tests [6] to full vehicle wind tunnel tests [7]. Even still, ground test setups can give rise to values different than those actually seen in real flight.

Whether using a canned simulation model or determining parameters via detailed experimentation, the difference between the parameter estimates and the actual parameters become sources of error in the flight dynamics model which can lead to suboptimal controller design. The methodology described herein for developing highly accurate flight dynamics models of quadrotor UAVs is done by applying in-flight sine wave injection into all critical control axes. Our iterative approach entails first using parameter estimates gained from bench testing to derive a best-guess dynamics model and then to use information learned from flight testing to hone in on the exact system parameters. This iterative approach allows for the development of highly accurate flight dynamics models and consequently a well-tuned flight controller.

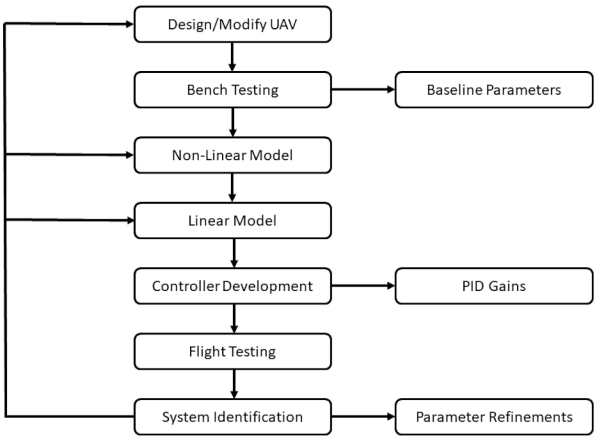

Fig. 1 Proposed System Identification Methodology.

Flight controller design techniques for UAVs are well established [8] and due to their graphical nature and ease of use, simulation programs such as Matlab and Simulink have been used extensively for developing models and controllers of various air vehicles [9]. Many researchers integrate Simulink with flight simulators such as X-Plane as mentioned above in order to better validate their flight controllers [10]. The problem with many of these simulations is that they do not capture physical limitations such as update rates, memory size, fixed word length effects, etc. of the hardware that actual flight controllers are deployed to. A way to simulate these real world phenomena is to employ hardware-in-the loop techniques which entail generating C code from a Simulink controller model that can then target hardware systems such as xPC target [11]. It is also possible to have the best of both worlds where hardware with auto-generated C code from Simulink can communicate directly to an X-Plane simulation [12] [13]. Even though targeting a flight controller to hardware is one step closer to the desired system, a simulated dynamics model still suffers from parameter uncertainty. The best way to accurately determine model parameters is to extract them from flight test data. Instead of relying on a simulation program such as X-plane linked to a flight controller, the Fusion 1 UAV used for this research contains a flight controller composed of code auto-generated from Simulink. This hardware topology gives one the capability to

transition between traditional simulation mode and full experimentation mode where the auto-coded flight controller and UAV physics are acting together in real time. Using model-based design throughout the entire system identification work flow has also allowed for the development of a Simulink-based flight controller and optimization of gains therein.

In this paper, we explore the use of a motion capture system to accurately characterize the dynamic behavior of a small (250 g, 180 mm) quadrotor drone. We start with the design and characteristics of the drone, experimental setup, simulation and modeling, control loop design, and finally cover flight testing under manual and autonomous control via the motion capture system, which was used as both a data collection device and, ultimately, the primary feedback sensor for the position and velocity loops.

III. Drone Characteristics

The Fusion 1 drone by Craft Drones was used for all of the flight testing in this paper. Like most drones, the Fusion 1 has a core sensor suite consisting of inertial measurement units (IMUs) which are used by the angle and rate loops as well as a bus voltage sensing analog to digital converter (ADC) which is used for bus voltage compensation. All communication with the drone was done either via the Spektrum transmitter, which pairs with the on-board Spektrum receiver, or via a WiFi connection with the host PC. The Fusion 1 drone also has a barometer which was not used for our experiments. Unlike many traditional drones the Fusion 1 drone is based upon the Snickerdoodle System on Module (SOM) [14] that has an FPGA System on Chip (SoC) as its main processing system which allows for unique algorithm segmentation between the CPUs and FPGA fabric. The flight controller was coded in Simulink and targeted to the drone processing cores via Simulink External Mode. This model-based design and deployment capability greatly shortened development times and allowed for real-time signal logging and visualization at update rates of 1 kHz.

Table 1 Fusion 1 Parameters.

Parameter	Value
Chipset	Xilinx Zynq 7020
CPU	32-bit Dual-Core ARM Cortex-A9
FPGA	1.3M Gates/53,200 LUT-6
WiFi	150Mbps 2×2 MIMO 2.4GHz/5GHz 802.11n

Fig. 2 Fusion 1 Drone from Craft Drones. Craft Drones LLC.

Some other features of the Fusion 1 drone depicted in Fig. 2 are the landing pads and prop guards, which kept the drone and researchers safe. Also shown, but not used for this article, is the global shutter camera which is directly connected to the FPGA fabric allowing for future real-time image processing for autonomous visual servoing of the aircraft [15] [16]. As mentioned above, a Spektrum transmitter is used to control the drone and the custom control surface mappings of the transmitter are shown below in Fig. 3. The drone can be operated in manual, autonomous, or semi-autonomous modes, all of which are discussed in Section III.

Fig. 3 Fusion 1 Transmitter Mapping. Craft Drones LLC.

For power and actuation the Fusion 1 drone has EMAX 15 AMP ESCs and 4500 KV 1106 BLDC motors. The props are 3 bladed with a diameter of 2.3 inches and a pitch of 4.5 inches (2345). The thrust and torque curves of the entire ESC, motor, propeller ensemble were determined experimentally with an RC Benchmark dynamometer [17] and are shown below in Fig. 4. The RC Benchmark tool contains an optical sensor that determines precise motor speed and strain gauges that measure the propeller thrust and torque.

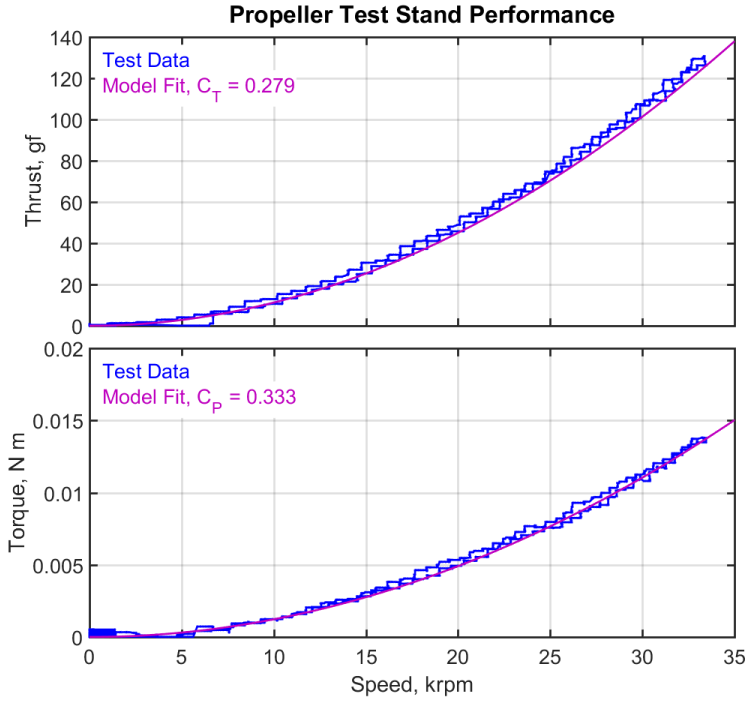

Fig. 4 Propeller Performance Curve.

Table 2 lists all of the physical parameters characterizing the Fusion 1 drone in flight. Some parameters, such as mass and geometry, can be determined easily via direct measurement, whereas other parameters, such as propeller coefficients and moments of inertia, can only be determined with special equipment. Flight tests were used to obtain accurate values for the last six parameters listed in Table 2.

Table 2 Fusion 1 In-Flight Parameters.

Parameter	Value	Source
D_r	0.0584 m	Measured/purchase specification
L_x	0.0635 m	Measured/purchase specification
L_y	0.0635 m	Measured/purchase specification
m	0.250 kg	Measured/purchase specification
K_e	0.0021 V/(rad/s)	Motor specification
K_t	0.0021 N m/A	Motor specification
V_{batt}	11.1 V	Measured/purchase specification
R_{tt}	0.269 ohm	Measured with multimeter
C_T	0.279	Measured with dynamometer
C_P	0.333	Measured with dynamometer
g	9.80665 m/s²	Gravity constant
ρ	1.204 kg/m³	Standard atmosphere at test locale [18]
$B_{m_{hover}}$	5.47×10^{-6} N m/(rad/s)	Calculated
$\omega_{m_{hover}}$	2.49×10^{3} rad/s	Calculated
I_{xx}	4.27×10^{-4} kg m²	Estimated, refined from flight tests (dc gain of ω_x / roll cmd)
I_{yy}	6.09×10^{-4} kg m²	Estimated, refined from flight tests (dc gain of ω_y / pitch cmd)
I_{zz}	1.50×10^{-3} kg m²	Estimated, refined from flight tests (dc gain of ω_z / yaw cmd)
J_m	6.45×10^{-7} kg m²	Estimated, refined by flight tests (break frequency of all inner loops)
T_d	0.005 s	Estimated, refined in flight tests (phase of frequency responses)
T_{d_2}	0.030 s	Estimated, refined in flight tests (phase of frequency responses)

IV. Experimental Setup

Complete characterization of the physics of a small air vehicle is limited by the quality of the sensor data comprising rates, angles, velocities and positions. While small, inexpensive MEMS IMUs are commercially ubiquitous and provide a robust reading of body rates and accelerations, other states are harder to obtain. Pitch and roll Euler angles are usually obtained from the gravity angle sensed by the IMU along with integrals of the body rates, using a complementary or Kalman filter [19]. Heading angle likewise can be sensed, imperfectly, from a magnetic compass, altitude from a barometric altimeter, and velocities and positions from a GPS.

In order to completely characterize the plant dynamics, it is necessary to know the exact position of the UAV in 3D space in real time. When flying outdoors, GPS can be used to provide positional feedback to the autotuning algorithm. Flying indoors presents major obstacles in determining UAV position however, as GPS works poorly or not at all. Other sensors such as magnetic compasses can be affected by nearby metal structures, and miniature barometric sensors are sensitive to small pressure changes and are not absolute in position. For our purposes, an ideal solution for GPS-denied indoor position determination exists in the form of a commercial motion capture system.

Motion capture systems, which are commonly used in the video game and film industries for superimposing computer generated imagery (CGI) effects onto actors, use multiple ceiling-mounted infra-red cameras to detect and precisely locate objects within their field of view. Such a system can track a small indoor drone, equipped with reflective markers, to an accuracy better than one millimeter as well as a fraction of a degree in angle, at one hundred frames per second or higher. Test command inputs are injected into the UAV loops and the precise position of the UAV is determined by the absolute tracking system and recorded in real time.

Fig. 5 shows the hardware topology used for all the flight experiments mentioned in this article. The four high speed tracking cameras send data to the host PC via ethernet where the independent data streams are combined and converted into a single 3D position and orientation of the drone. This drone pose data is then sent to the drone via WiFi where it is used to close the output position loops. All internal flight controller state data is monitored over WiFi by Simulink External Mode which is running on the host computer. This topology allowed for the rapid deployment of new controllers onto the drone by simply pressing a single "play" button which compiled the Simulink flight controller

model into C code and sent it wirelessly to the drone where it ran in real time.

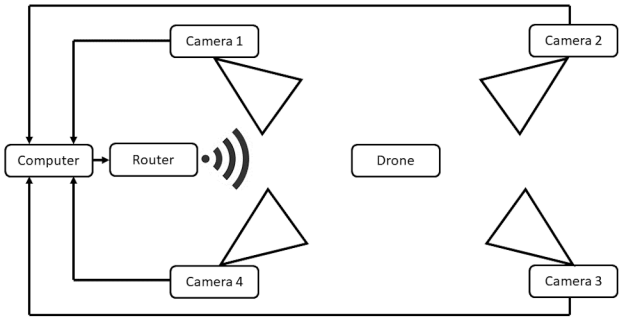

Fig. 5 Experimental Setup.

V. Linear Model

In this section, the linear models characterizing the airframe, and used as a basis for designing the control loops, are derived. Though high-order state space formulations are popular ([20] for example), we simplify the model to a rigid-body, decoupled, symmetric airframe, linearize the system about hover, and ignore body aerodynamics. This simplification allows some insight into the transfer function properties and relevant parameters. However, for a quadrotor drone of any size, the motor dynamics are key and *cannot* be neglected. Accounting for phase delay due to update rate, buffering and sampling in a digital feedback control loop is also critically important.

The thrust and torque of a rotor are classically characterized as proportional to rotor speed squared (Fig. 4) and nondimensionalized using thrust and power coefficients [21]:

$$T = \frac{\rho C_T D_r^4}{(2\pi)^2}\omega_m^2 \tag{1}$$

$$Q = \frac{\rho C_P D_r^5}{(2\pi)^3}\omega_m^2 \tag{2}$$

where motor speed ω_m is in radians per second. The square-law behavior was well-verified in our propeller bench tests (Fig. 4). For hover and moderate forward speeds, we only need to be concerned about static thrust and can ignore advance ratio effects. For a quadrotor drone, the trim motor speed at hover which generates enough net thrust to support the weight of the vehicle is

$$\omega_{m_{hover}} = \sqrt{\frac{mg}{4\rho C_T D_r^4 (\frac{1}{2\pi})^2}} \tag{3}$$

A simple but physically representative motor/prop dynamic model [22] comprising voltage, resistance, torque and back emf constants, and inertia is given by

$$I_m R_{tt} = V_{batt} f_{dc} - K_e \omega_m \tag{4}$$

$$J_m \omega_m s = K_t I_m - B_m \omega_m \tag{5}$$

where J_m is the inertia of the propeller (plus the motor armature, but it is usually insignificant compared to the propeller). Motor inductance is also negligible for small outrunner drone motors; typically in the microhenries. K_t and K_e are equal in SI units and are both equal to $\frac{60}{2\pi}\frac{1}{K_v}$, where K_v is the commonly advertised voltage constant for hobbyist drone motors, in rpm/volt. f_{dc} is the fraction of duty cycle command given to the motors, which are generally pulse-modulated. The damping coefficient, B_m, must be calculated at the hover trim condition from the propeller torque:

$$B_m = \left.\frac{dQ}{d\omega_m}\right|_{hover} = \frac{2\rho C_P D_r^5}{(2\pi)^3}\omega_{m_{hover}} \tag{6}$$

If the motors are commanded symmetrically, the tilt response of the rigid-body vehicle to a delta change in thrust (two lower on one side and two higher on the other) is

$$I_{xx}\dot{\omega}_x = 4\Delta T L_y \tag{7}$$

where L_y is the lateral distance from the center of mass (assumed symmetric) to the prop hubs, in a standard rectangular arrangement. The transfer function describing the roll rate response of the drone to a motor command distributed appropriately to all four motors is:

$$G_{f_{dc}}^{\omega_x} = \left(\frac{K_t V_{batt}}{J_m R_{tt} s + K_t K_e + B_m R_{tt}}\right)\left(\frac{\sqrt{mg\rho C_T D_r^4}}{(2\pi)}\right)\left(\frac{4L_y}{I_{xx}s}\right)e^{-T_d s} \tag{8}$$

This expression also includes calculation and sampling delays in the loop, T_d, which we represented with a 2nd-order Padé approximation. We have ignored cross-axis coupling, and aerodynamic forces due to sideslip and rotation and center-of-mass-offsets, but the resulting transfer function–a first order lag, an integrator, and a delay–is adequate to accurately characterize the drone and facilitate control loop gain selection.

An important detail is that the battery voltage may drop precipitously during flight. Since V_{batt} has a directly proportional effect on the loop gain of the system, the battery voltage should be monitored and compensated for by increasing the loop gain proportionally as V_{batt} decreases.

The transfer function for pitch is similarly:

$$G_{f_{dc}}^{\omega_y} = \left(\frac{K_t V_{batt}}{J_m R_{tt} s + K_t K_e + B_m R_{tt}}\right)\left(\frac{\sqrt{mg\rho C_T D_r^4}}{(2\pi)}\right)\left(\frac{4L_x}{I_{yy}s}\right)e^{-T_d s} \tag{9}$$

Of the other two inner loop control axes, vertical speed applies the thrust of all four motors to the drone mass, and has the same dynamic rolloff as the tilt loops:

$$G_{f_{dc}}^{\dot{h}} = \left(\frac{K_t V_{batt}}{J_m R_{tt} s + K_t K_e + B_m R_{tt}}\right)\left(\frac{\sqrt{mg\rho C_T D_r^4}}{(2\pi)}\right)\left(\frac{4}{ms}\right)e^{-T_d s} \tag{10}$$

It is sometimes assumed that a drone produces a yawing moment via the props pushing against the air differentially, but that is not quite true. Actually the motor stators push electromagnetically against the motor armatures and prop inertias, which push against the air. This subtle difference results in an extra lead in the yaw response transfer function:

$$G_{f_{dc}}^{\omega_z} = \left(\frac{K_t V_{batt}(J_m s + B_m)}{J_m R_{tt} s + K_t K_e + B_m R_{tt}}\right)\left(\frac{4}{I_{zz}s}\right)e^{-T_d s} \tag{11}$$

which makes the yaw loop inherently much easier to stabilize than the pitch and roll loops.

In hover, and neglecting body drag, the relationship between tilt angles and horizontal velocity of the CG is:

$$m\dot{u}_{x_{cg}} = F_x = -T\sin\theta \approx -mg\theta \tag{12}$$

$$m\dot{u}_{y_{cg}} = F_y = T\sin\phi \approx mg\phi \tag{13}$$

The sensed velocity/position location is not necessarily at the CG. It is approximately in the center of the planform but may be above or below the CG. The CG is offset from the reference position by an distance z_{cg}, approximately 2 cm in our case as the OptiTrack reflector markers are raised slightly on top and the battery is strapped to the bottom. Converting to the sensed position/velocity location:

$$u_x = u_{x_{cg}} - \dot{\theta} z_{cg} \tag{14}$$

$$u_y = u_{y_{cg}} + \dot{\phi} z_{cg} \tag{15}$$

The transfer function from the tilt angles, θ and ϕ, to sensed body-axis velocity is then (including sensor delay)

$$G_{\theta}^{u_x} = -\left(\frac{g}{s} + s\theta z_{cg}\right)e^{-T_{d2}s} \tag{16}$$

$$G_{\phi}^{u_y} = \left(\frac{g}{s} + s\phi z_{cg}\right) e^{-T_{d2}s} \tag{17}$$

The terms representing the difference between sensed velocity and CG velocity create a dramatic anti-resonance, which we will see in testing, and in fact, could probably be used to accurately measure the CG height. At the anti-resonance frequency, the drone rotates exactly about the sensor location, which remains fixed in space. Note the delay term T_{d2} is not necessarily the same as the delay in Eqs. 8-11, because the update rate from the OptiTrack, GPS or other position sensor can be slower than the gyro.

We neglect aerodynamic moments such as M_u, L_v, etc., permissible if the approximate center of pressure of the drone in lateral motion is the same height as the CG. Otherwise, a feedback exists from horizontal velocity to tilt moments, which can be stabilizing or destabilizing, and accounts for the common but incorrect belief that a high or low CG provides "pendulum" stability. For a drone with a fairly flat cross-section such as ours, we found these terms can be neglected.

VI. Control System Design

Our approach to the control system design used classical, decoupled, approximate airframe transfer function methods [23] to ensure adequate gain and phase margin in each loop. The plant transfer functions above for roll and pitch rate and altitude rate at hover are all of the form

$$G_v = \frac{K}{(s + a_m)s} e^{-T_d s} \tag{18}$$

where a_m is the motor/prop rolloff frequency and is equal to

$$a_m = \frac{K_t K_e + B_m R_{tt}}{J_m R_{tt}} \tag{19}$$

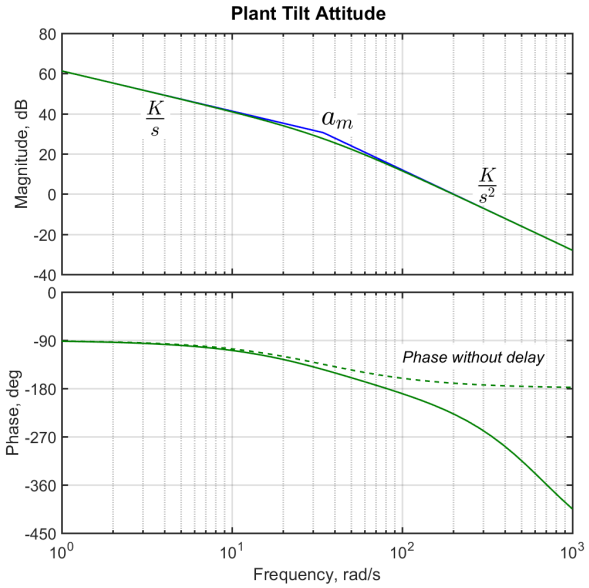

Fig. 6 Linear Model Plant Bode Plot. (Roll; Pitch and Altitude Rate Are Similar)

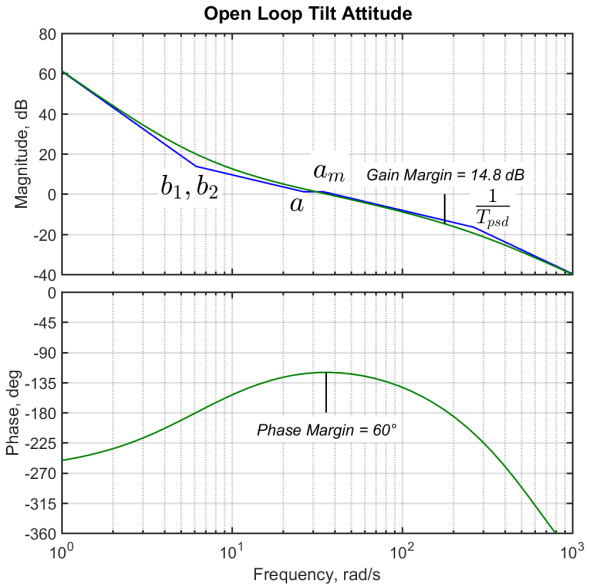

Fig. 7 Linear Model Open Loop Bode Plot. (Roll; Pitch and Altitude Rate Are Similar)

It is shown in the Bode plot of Fig. 6, which is also the principle transfer function to be identified in flight testing. This is amenable to a control architecture consisting of PID compensation on the inner loop (the gyro and altitude rate), and proportional gain on angle and altitude.

$$G_{open} = (K_p + s)\left(K_v + \frac{K_{v_i}}{s} + \frac{K_{v_d}s}{(T_{psd}s + 1)}\right)\frac{G_v}{s} \tag{20}$$

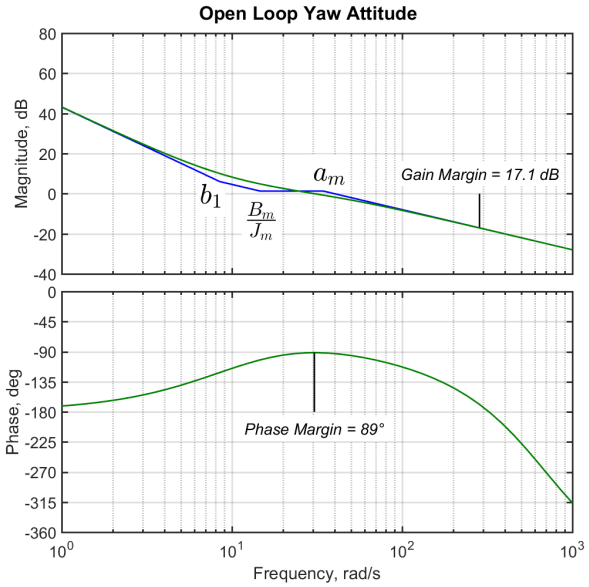

Fig. 8 Linear Model Open Loop Bode Plot. (Yaw)

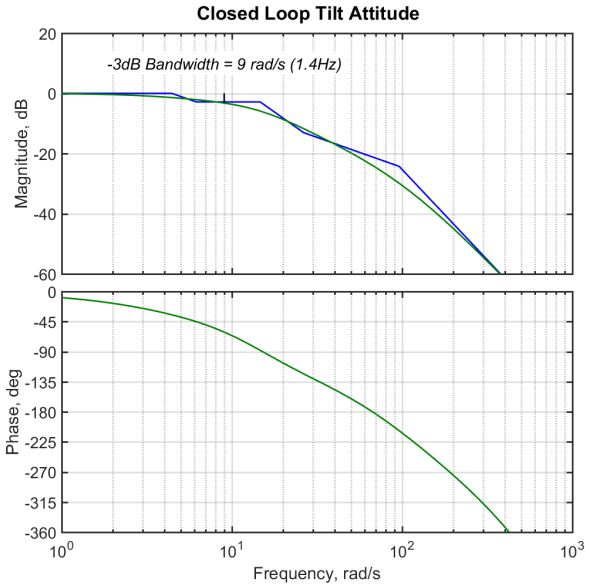

Fig. 9 Linear Model Closed Loop Bode Plot. (Roll; Pitch and Altitude Are Similar)

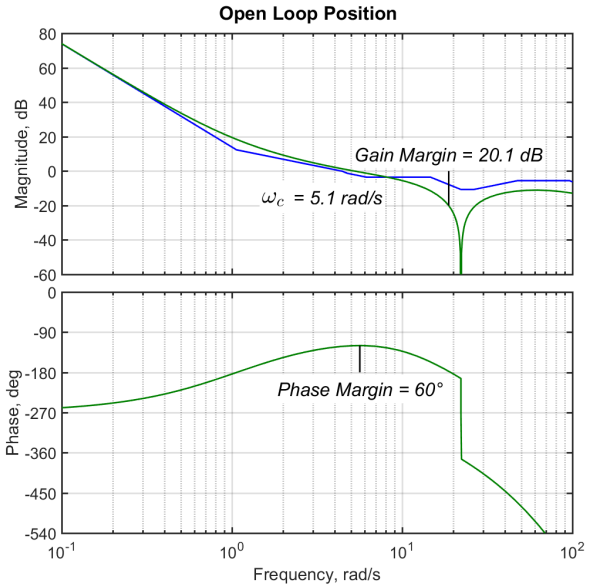

Fig. 10 Linear Model Open Loop Bode Plot. (Lateral Position)

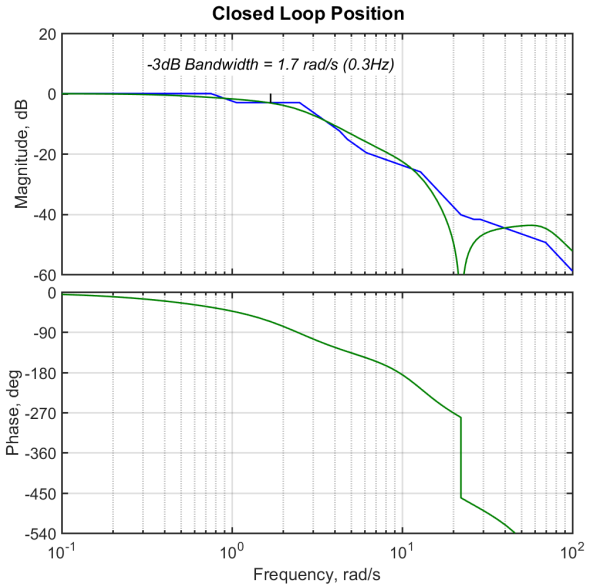

Fig. 11 Linear Model Closed Loop Bode Plot. (Lateral Position)

which is equivalent to

$$G_{open} = \frac{k(s+b_1)(s+b_2)(s+a)}{(T_{psd} * s + 1)s} \frac{Gv}{s} \tag{21}$$

An effective strategy is then to set $a = a_m$ to cancel the motor pole as closely as possible (though it doesn't have to be exact) and the pseudo-derivative pole $1/T_{psd}$, necessary for noise rejection, a factor of 10-20 higher. This extends the allowable crossover frequency and allows correspondingly higher bandwidth for the angle and altitude loops. The two zeros b_1 and b_2 can be set equal and as high as possible such that there is still a maximum of 60° of phase margin available. The gain is chosen such that the 0 dB crossover is at that frequency, satisfying McRuer's Crossover Law [24]. The system open loop Bode plot is shown in Fig. 7. The angle and PID gains are back-calculated algebraically from this

and result in reliably stable and high-performance loops for the pitch and roll angles, as well as altitude. The closed loop response, which predicts a bandwidth of 9 rad/s is shown in Fig. 9.

The heading (yaw) loop is simpler and has an extra lead frequency (Fig. 8). A similar loop closure applies, though the main concern is limiting the bandwidth so that the response doesn't drive sudden altitude disturbances.

The tilt (pitch and roll) loops constitute the main challenge of stabilizing the quadrotor. It remains to close the horizontal position loops, which can be done by a pilot or automatic controller with a position and velocity feedback.

The open-loop transfer functions for the position loop closures are then

$$G_{open}^{pos_x} = G_{ctrl} G_{\theta_{cmd}}^{\theta} G_{\theta}^{u_x} \frac{1}{s} \tag{22}$$

$$G_{open}^{pos_y} = G_{ctrl} G_{\phi_{cmd}}^{\phi} G_{\phi}^{u_y} \frac{1}{s} \tag{23}$$

where G_{ctrl} is the same structure, a PID on velocity and proportional gain on position, as in Eqs. 20 and 21; and is optimized the same way, only this time with linear position and velocity rather than angular, and with the -6dB bandwidth of the closed-loop angle response substituting for the motor pole a_m. Additionally, care must be taken to close the position loops at a bandwidth lower than the antiresonance given by Eqs. 16 and 17, as illustrated in Fig. 10. An alternative might be to get rid of the antiresonance by calculating a virtual sensed position coincident with the CG, but a crossover frequency of 5.1 rad/s is still achievable, which leads to a position closed-loop bandwidth of 1.7 rad/s (Fig. 11).

VII. Nonlinear Time Domain Modeling and Simulation

A nonlinear time-domain simulation model was constructed in Matlab/Simulink and is shown in Fig. 12. This uses standard rigid-body equations of motion [25]

$$\frac{d}{dt} u_{cg} = \frac{F}{m} + u_{cg} \times \omega \tag{24}$$

$$\frac{d}{dt} \omega = I^{-1} (I\omega \times \omega + M_{cg}) \tag{25}$$

where I is the 3×3 inertia matrix (diagonal, the products of inertia are assumed negligible due to symmetry), and F and M_{cg} are the net forces and moments on the vehicle, from aerodynamics, gravity and landing contact. The model includes quaternion representation of attitude, provision for center-of-mass offsets, sensor offsets, misalignments and lags; body drag, wind, gusts, and gravity. Using vectorized signals and matrix parameters, any number of motors and props at any position and orientation can be simulated, e.g., octorotors. We also calculate forces and moments due to landing leg contact with the ground plane, allowing accurate simulation of takeoff and landing. Optionally constraining motion solely to rotation simulates operation in a gimbal test fixture.

Importantly, a detailed model of the electric motor dynamics was built into the model, with classical square-law representation of the propeller thrust and drag with motor speed. A battery model with discharge curve was also included, with bus voltage compensation in the inner control loop gains. Digital update delays of 1 ms were included on the input and output of the controller which simulate the real world controller data update time.

The model also features 3-D animation of the drone's position, attitude and motor speeds using Matlab graphics, and joystick control input allowing a user to perform a virtual test flight in real time (Fig. 13). If being controlled manually, the four joystick inputs map to the pitch and roll Euler angle commands, yaw rate command and vertical speed. In autonomous mode, the vehicle plant model (or the external motion capture system in the hardware) provides feedback of position, velocity and heading, allowing outer loops to be closed and the vehicle to follow pre-determined mission paths.

The controller model incorporates classical, hierarchical feedback loop closures starting with body-axis gyro feedback, mapped to the four motor commands in the architecture described in Section VI. The nonlinear model was used to test the gains derived via linear analysis prior to testing them in flight. PID control is used in the inner gyro loops, and the translational velocity loops. The integral component of those is instrumental in compensating for center-of-mass imbalance, and wind, respectively.

In the hardware, Euler angle feedback can be obtained either from a complementary filter on the IMU's gyro and accelerometer, or directly from the external motion capture system, and the simulation allows either option for comparison.

Lastly, provision was made for automatic injection of a customizable sequence of sinewaves into any of the lift, roll, pitch or yaw axes for dynamic characterization. This was a precursor to doing the same thing in flight tests. The responses of the nonlinear model were found to agree very closely with the linear models, implying very little cross-coupling and nonlinearity, at amplitudes as high as 10 percent of the maximum motor commands. This helped build confidence in the control design leading up to flight testing.

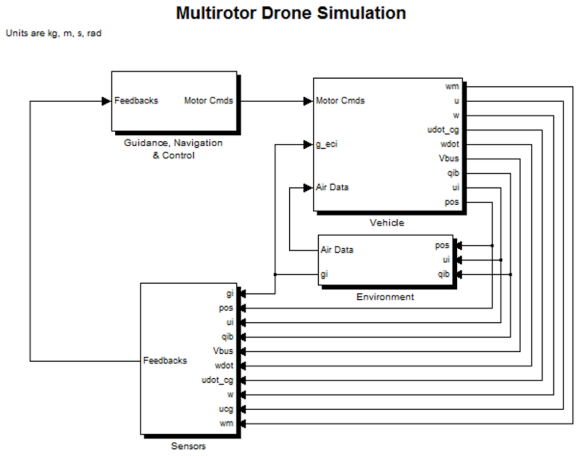

Fig. 12 Top-Level Simulation Model.

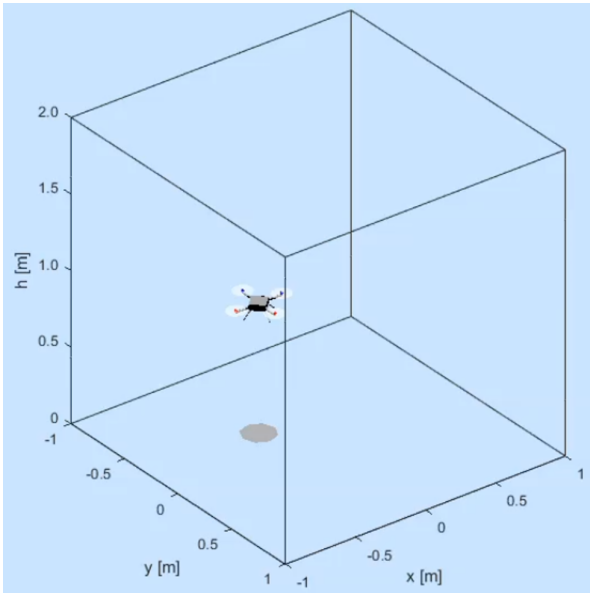

Fig. 13 Real-Time Pilot-in-the-Loop Animation.

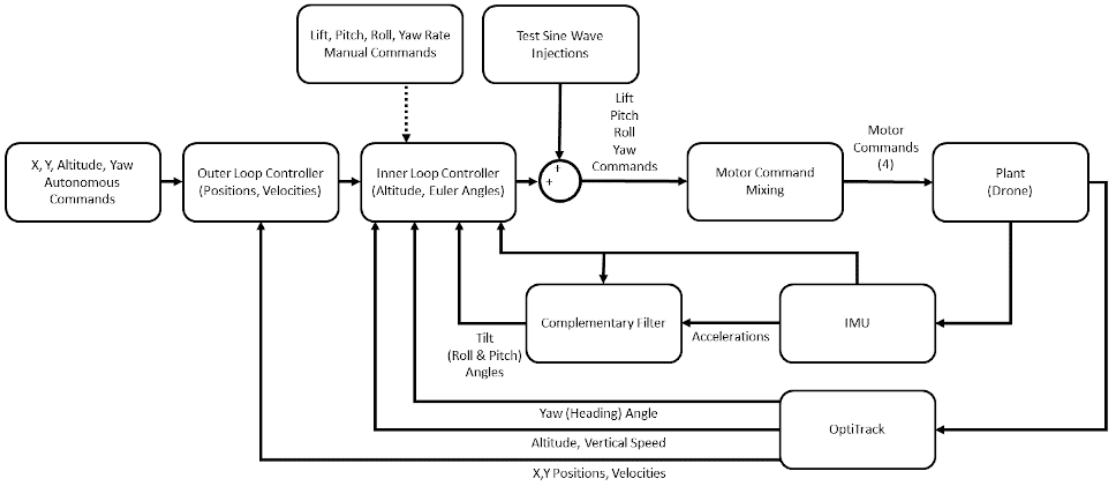

Fig. 14 Controller Block Diagram Showing Test Signal Injection Point.

VIII. Flight Tests and System Identification

A number of flight test sessions, of increasing complexity, were made during the course of the drone controller development. Initially, the drone was controlled using only proportional gains on gyro, angle and vertical speed feedback. This was facilitated by the OptiTrack system, which provided the luxury of precise real time measurements of the angles without complimentary or Kalman filtering of the IMU. In this way, in spite of having imprecise estimates of several key parameters, it could be made stable enough to fly manually via transmitter command. Once in the air, test signals could

be injected while the pilot stabilized the vehicle and kept it in bounds. Figure 14 shows the overall control and test architecture and where the test signals were injected.

This allowed increasingly precise characterization of the system, especially the vehicle and propeller moments of inertia, the propeller characteristics, and the loop update delays (Table III). As knowledge of the system was refined, and various configuration changes emerged such as using a larger battery and different props, the linear model in Section V and the nonlinear model in Section VII continued to be developed in parallel. Eventually, the outer loops (altitude, heading and horizontal position and velocity) were closed, allowing fully autonomous flight, and features such as noise filtering, anti-windup integrators, full PID compensation, angles from complementary filtering IMU rates and accelerometer signals, battery voltage droop compensation, takeoff and landing maneuvers, and trajectory following were added.

We continued to use sine wave injection as the primary system identification mechanism. Though many investigators use chirps or doublet inputs ([26] and [27] for example), discrete sinewaves have some advantages: they are simple to implement and simple to process, avoid problems with insufficient dwell time, and allow detailed examination of waveforms in the data, which may more easily reveal nonlinearities such as saturation, rate limiting or deadbands.

The sine input command used as a stimulus for drone system identification is shown in Fig. 15. The injection begins 20 seconds into the flight, to give the drone enough time to autonomously take off and achieve a stable hover at 1 meter altitude. The sine waves are injected into the motor commands in the axis to be tested (roll, pitch, yaw or lift). The amplitudes increase with frequency to compensate for the rolloff of the dynamic response and result in a roughly consistent output amplitude. Additionally, to assess the linearity of the system, we ran tests multiplying all the amplitudes by factors of two and four.

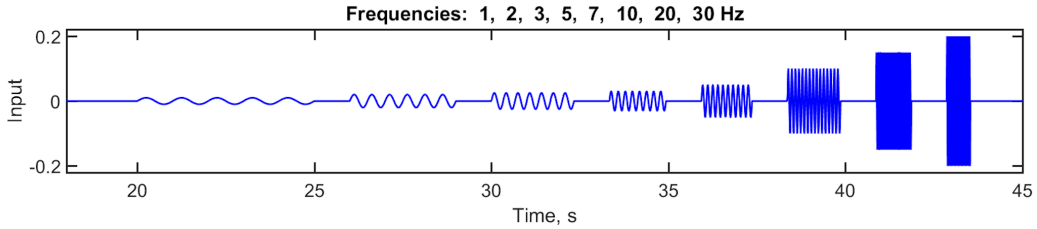

Fig. 15 Sine Wave Injection Signal.

Comparisons of the flight test dynamic responses in all four axes with the linear model are shown in Figures 16 through 21. The results of the nonlinear Simulink model described in Section VII, run with sine injections in exactly the same way as in flight, are also plotted.

The model match all around is very satisfactory, indicating a thorough capturing of the system dynamics in hover. This is in spite of having made a number of dramatic simplifying assumptions in the linear model: rigid body, decoupled axes, symmetric, basic motor and prop models, and no body drag or lift aerodynamics. Additionally, there is very little spread in the points over 1x, 2x and 4x input amplitudes, showing that the drone's behavior is essentially linear (An exception is the data points at 1 Hz in Fig. 17. This occurred because the test injection signal in this run was very nearly cancelled by the controller, causing the input to be lost in the noise).

The motor break frequency, the low frequency K/s asymptotes, and the extra phase lag due to sampling are all easily identifiable. The roll and pitch response magnitudes are slightly different (Figs 16 and 17), due to the minor difference in roll and pitch moments of inertia (which is mostly because the battery is rectangular). The extra lead in the high-frequency yaw rate response (Fig. 19) is apparent. We used these results to modify initial estimates of these parameters and re-optimize the controller gains per Section V. Figs. 20 and 21 show the response of speed to tilt angle, and dramatically illustrate the antiresonance caused by the 2 cm offset between the CG and the OptiTrack-sensed virtual position.

This accurate characterization of the system, along with the excellent performance of the on-board IMU and OptiTrack sensors, allowed very tight, high-bandwidth control of the drone in flight. We flew the drone from takeoff, through a square pattern one meter on a side, and back to landing as a standard flight test. Figure 22 compares the results of this flight before and after optimization of gains. In particular, the earlier flight only used proportional-integral control without the derivative term, which we were able to add after a precise characterization of the motor rolloff and sensor lags. Figure 23 shows the same flight test data comparison in the form of time histories.

The final flights were extremely stable, and in hover, exhibited a position holding accuracy of approximately one

centimeter in horizontal position and two millimeters in altitude, with respect to the OptiTrack position feedbacks. This was despite taking place in a fairly small (4 x 4 m) laboratory room, with strong recirculating air currents.

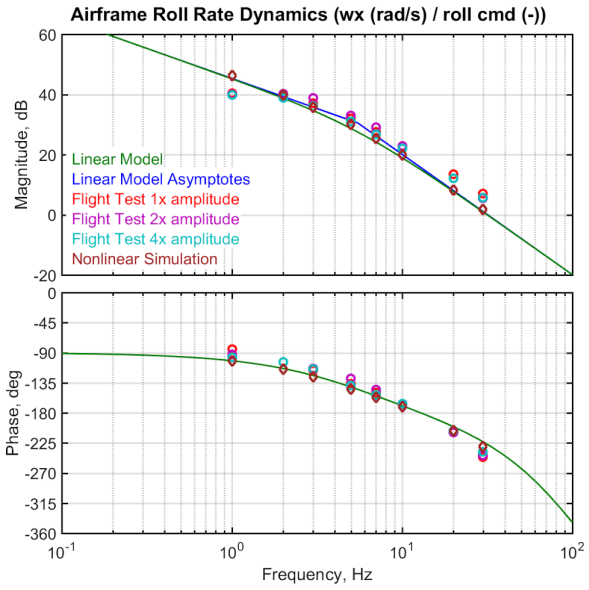

Fig. 16 Roll Rate Response.

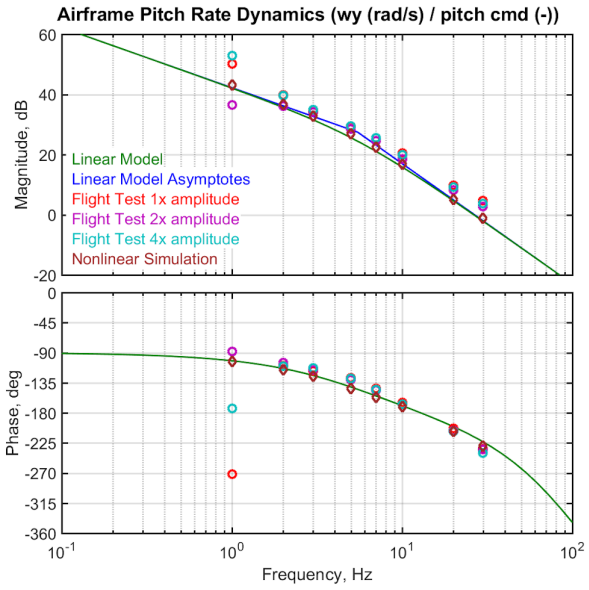

Fig. 17 Pitch Rate Response.

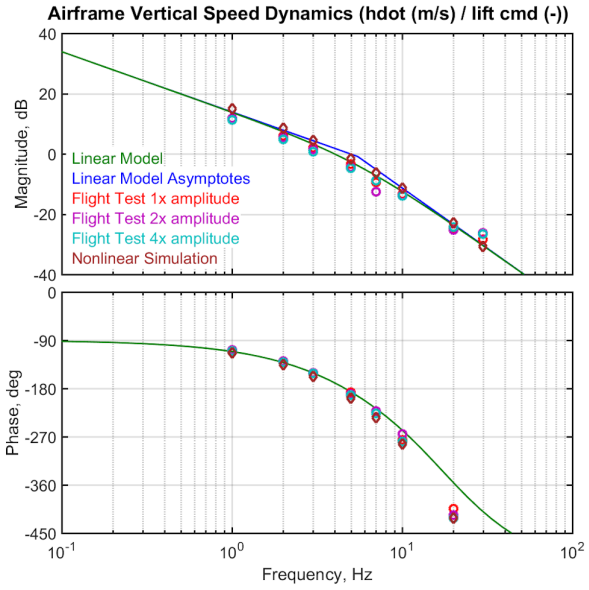

Fig. 18 Vertical Speed Response.

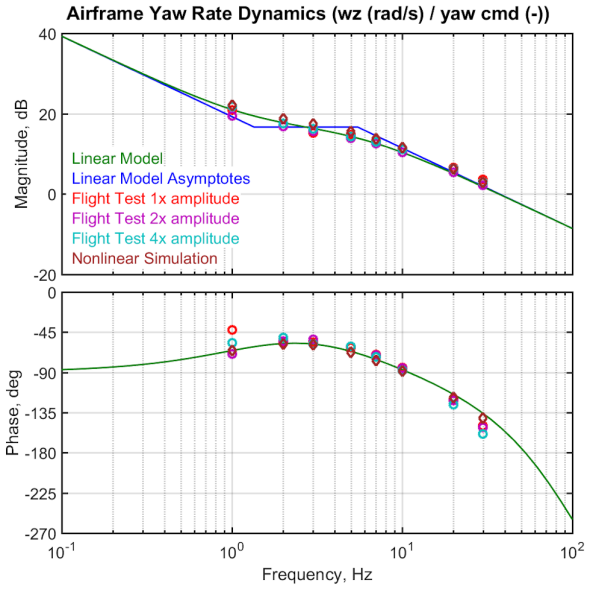

Fig. 19 Yaw Rate Response.

IX. Conclusion

In this investigation, we were able to comprehensively determine a complete physics model of a small indoor drone, and use that to optimize a 6-DoF control loop architecture for completely autonomous flight. This was using a simple linear transfer function approach with a minimalist set of assumptions, with rigid-body, decoupled axes, and no body aerodynamics. However, motor and prop dynamic responses, and update lag in the loops were critically important. Using sine wave inputs at discrete frequencies, we have shown that the linear model matches the data in all the inner

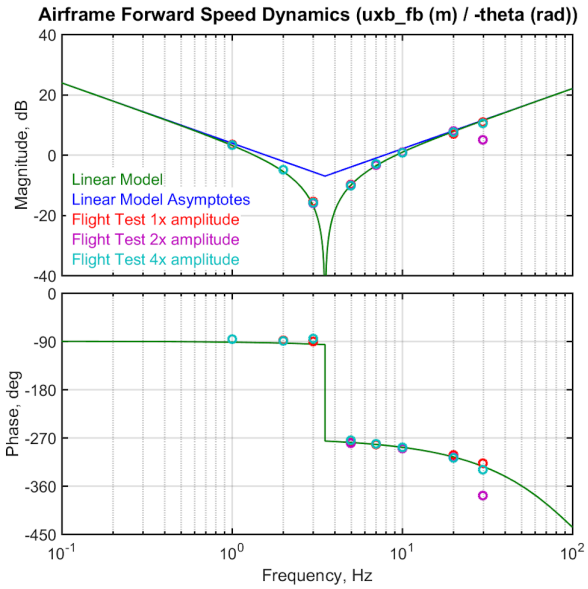

Fig. 20 Forward Speed Response.

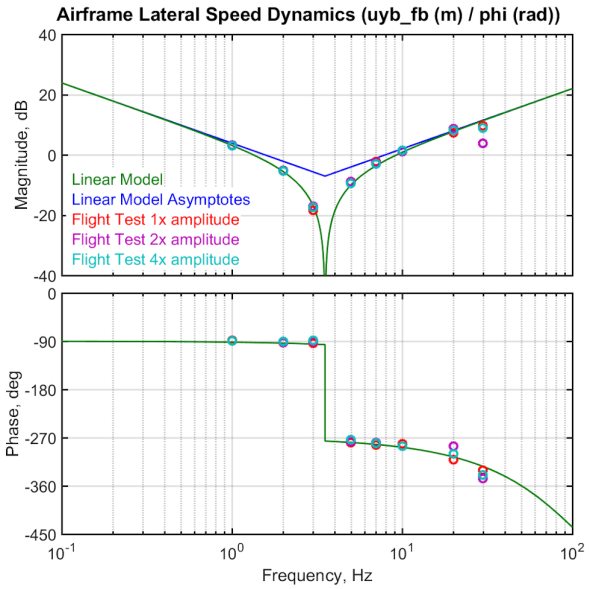

Fig. 21 Lateral Speed Response.

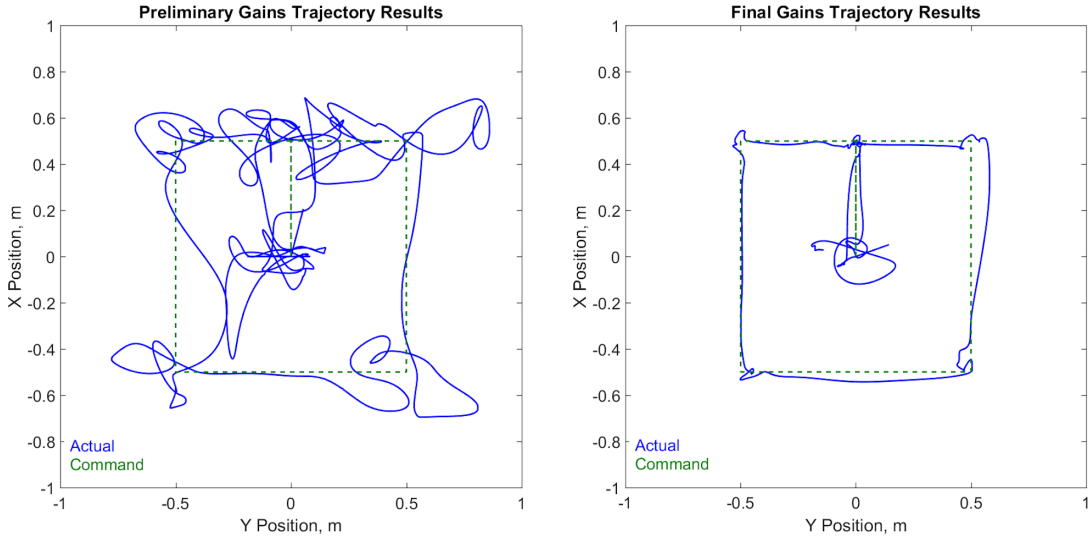

Fig. 22 Flight Trajectory Before and After Gain Optimization.

and outer loop responses, as well as the nonlinear simulation model. The optimized controller was able to produce extremely stable autonomous indoor flight, accurately following a pre-programmed trajectory.

Future plans for the system identification framework include extending it to other flight regimes beyond hover, including high speed forward flight, climb, descent, banked turns, formation following, and different CG configurations. To this end, outdoor autonomous flight could be accomplished with the aid of GPS and appropriate Kalman filtering in place of the OptiTrack cameras. Another focus area is to replace the OptiTrack system with a calculation of absolute pose using the on-board camera of the Fusion 1, with the aid of wall-mounted fiduciary tags or other visual cues in the environment. Establishing a solid baseline dynamics model with representative simulation is key to these activities.

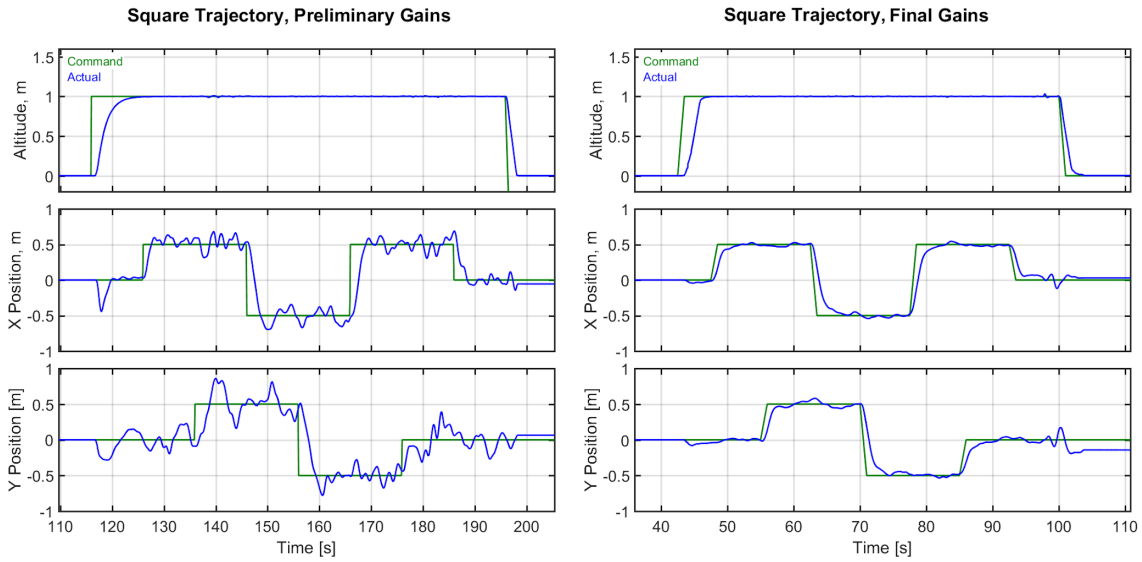

Fig. 23 Trajectory Time histories Before and After Gain Optimization.

X. Acknowledgements

Special thanks to the MathWorks and Moog for their continued help and support.

References

[1] Kaputa, D. S., Bauch, T., Roberts, C., McKeown, D., Foote, M., and Salvaggio, C., "MX-1: A New Multi-Modal Remote Sensing UAS Payload with High Accuracy GPS and IMU," *2019 IEEE Systems and Technologies for Remote Sensing Applications Through Unmanned Aerial Systems (STRATUS)*, 2019, pp. 1–4. doi:10.1109/STRATUS.2019.8713292.

[2] Sabikan, S., and Nawawi, S. W., "Open-source project (OSPs) platform for outdoor quadcopter," *Journal of Advanced Research Design*, Vol. 24, 2016, pp. 13–27.

[3] Sreeraj, A., Kaputa, D., and Crassidis, A., "A Model-Free Control Algorithm Based On The Sliding Mode Control Method With Applications to Unmanned Aircraft Systems," *A Model-Free Control Algorithm Based On The Sliding Mode Control Method With Applications to Unmanned Aircraft Systems*, 2019, p. 113. doi:10.11159/cdsr19.113, URL `http://avestia.com/CDSR2019_Proceedings/files/paper/CDSR_113.pdf`.

[4] Schulken, E., and Crassidis, A., "Model-Free Sliding Mode Control Algorithms including Application to a Real-World Quadrotor," *Model-Free Sliding Mode Control Algorithms including Application to a Real-World Quadrotor*, 2018, p. 112. doi:10.11159/cdsr18.112, URL `http://avestia.com/CDSR2018_Proceedings/files/paper/CDSR_112.pdf`.

[5] Ou, Q., Chen, X., Park, D., Marburg, A., and Pinchin, J., "Integrated Flight Dynamics Modelling for Unmanned Aerial Vehicles," *2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, 2008, pp. 570–575. doi:10.1109/MESA.2008.4735660.

[6] Gabriel, D. L., Meyer, J., and Plessis, F. d., "Brushless DC motor characterisation and selection for a fixed wing UAV," *IEEE Africon '11*, 2011, pp. 1–6. doi:10.1109/AFRCON.2011.6072087.

[7] Blake, W., Dickes, E., and Gingras, D., "UAV aerial refueling-wind tunnel results and comparison with analytical predictions," *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2004, p. 4820.

[8] Salih, A. L., Moghavvemi, M., Mohamed, H. A., and Gaeid, K. S., "Flight PID controller design for a UAV quadrotor," *Scientific research and essays*, Vol. 5, No. 23, 2010, pp. 3660–3667.

[9] Chen, C., and Ji, Y., "Modular Aircraft simulation platform based on Simulink," *2010 IEEE International Conference on Mechatronics and Automation*, 2010, pp. 1454–1459. doi:10.1109/ICMA.2010.5589162.

[10] Ribeiro, L. R., and Oliveira, N. M. F., "UAV autopilot controllers test platform using Matlab/Simulink and X-Plane," *2010 IEEE Frontiers in Education Conference (FIE)*, 2010, pp. S2H–1–S2H–6. doi:10.1109/FIE.2010.5673378.

[11] Lu, P., and Geng, Q., "Real-time simulation system for UAV based on Matlab/Simulink," *2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering*, Vol. 1, 2011, pp. 399–404. doi:10.1109/CCIENG.2011.6008043.

[12] Adiprawita, W., Suwandi Ahmad, A., and Sembiring, J., "Hardware in the Loop Simulation for Simple Low Cost Autonomous UAV (Unmanned Aerial Vehicle) Autopilot System Research and Development," *Hardware in the Loop Simulation for Simple Low Cost Autonomous UAV (Unmanned Aerial Vehicle) Autopilot System Research and Development*, 2012, p. 218.

[13] Ernst, D., "Development of Research Platform for Unmanned Vehicle Controller Design, Evaluation, and Implementation System: From MATLAB to Hardware Based Embedded System," *Graduate Theses and Dissertations*, 2007. URL https://scholarcommons.usf.edu/etd/699.

[14] "krtkl | Accelerated Intelligence," , 2019. URL https://krtkl.com/.

[15] Nonami, K., Kendoul, F., Suzuki, S., Wang, W., and Nakazawa, D., *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*, Springer Science & Business Media, 2010.

[16] Gans, N. R., Dixon, W. E., Lind, R., and Kurdila, A., "A hardware in the loop simulation platform for vision-based control of unmanned air vehicles," *Mechatronics*, Vol. 19, No. 7, 2009, pp. 1043–1056. doi:10.1016/j.mechatronics.2009.06.014, URL http://www.sciencedirect.com/science/article/pii/S0957415809001263.

[17] "RCbenchmark | Tools for Drone and Robot Designers," , 2019. URL https://www.rcbenchmark.com/.

[18] Administration, U. S. N. O. a. A., Administration, U. S. N. A. a. S., and Atmosphere, U. S. C. o. E. t. t. S., *U.S. Standard Atmosphere, 1976*, National Oceanic and Amospheric [sic] Administration, 1976. Google-Books-ID: x488AAAAIAAJ.

[19] Valenti, R. G., Dryanovski, I., and Xiao, J., "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs," *Sensors*, Vol. 15, No. 8, 2015, pp. 19302–19330. doi:10.3390/s150819302, URL https://www.mdpi.com/1424-8220/15/8/19302.

[20] Gong, A., Sanders, F. C., Hess, R. A., and Tischler, M. B., "System Identification and Full Flight-Envelope Model Stitching of a Package-Delivery Octocopter," *AIAA Scitech 2019 Forum*, 2019, p. 1076.

[21] Von Mises, R., *Theory of flight*, Courier Corporation, 1959.

[22] Hendershot, J. R., and Miller, T. J. E., *Design of brushless permanent-magnet machines*, Motor Design Books, 2010.

[23] McRuer, D. T., Graham, D., and Ashkenas, I., *Aircraft Dynamics and Automatic Control*, Princeton University Press, 2014. Google-Books-ID: bs__AwAAQBAJ.

[24] McRuer, D., and Jex, H., "A Review of Quasi-Linear Pilot Models," *IEEE Transactions on Human Factors in Electronics*, Vol. HFE-8, No. 3, 1967, pp. 231–249. doi:10.1109/THFE.1967.234304.

[25] Greenwood, D. T., *Principles of dynamics*, Prentice-Hall Englewood Cliffs, NJ, 1988.

[26] Schulze, P. C., Miller, J., Klyde, D. H., Regan, C. D., and Alexandrov, N., "System Identification of a Small UAS in Support of Handling Qualities Evaluations," *AIAA Scitech 2019 Forum*, 2019, p. 0826.

[27] Cho, S. H., Bhandari, S., Sanders, F. C., Tischler, M. B., and Cheung, K., "System Identification and Controller Optimization of Coaxial Quadrotor UAV in Hover," *AIAA Scitech 2019 Forum*, 2019, p. 1075.