AIAA Modeling and Simulation Technologies Conference and Exhibit
20 - 23 August 2007, Hilton Head, South Carolina

AIAA 2007-6569

Hardware-in-the-loop Simulation Design for Evaluation of Unmanned Aerial Vehicle Control Systems

Eric R. Mueller*
NASA Ames Research Center, Moffett Field, CA, 94035

This paper discusses the construction and testing of hardware-in-the-loop simulations using a commercial software simulation package and a custom-designed simulation. It discusses the process of integrating an avionics computer with Unmanned Aerial Vehicle (UAV) sensors and actuators, designing and implementing linear and non-linear simulations of the aircraft, setting up the control system architecture and evaluating various control laws through the hardware-in-the-loop simulations. Extensive comparisons are made between the different versions of the simulations to ensure that every step in the piecewise development of the final simulation is correct. Several types of control systems were tested on this final simulation. However, despite their adequate tracking of reference trajectories, none are robust and mature enough to yet consider for in-flight testing. A future work section discusses options for further development of the control system and modifications to the simulations to increase their fidelity. A straightforward, detailed and logical process is provided for setting up hardware-in-the-loop simulations of small UAV systems similar to the one described here, and evaluating control system performance. Important time and cost savings from lessons learned in this process are also provided.

I. Introduction

Unmanned Aerial Vehicle (UAV) systems have proliferated in recent years due to the falling costs of high powered, embedded computers, inertial measurement units (IMUs) and other electronic components. Real-time operating systems such as *VxWorks*, *QNX*, and *LinuxRT* provide software support for time-critical functions onboard the aircraft. Researchers with modest experience in hardware, software and control system design can build a UAV from scratch, although the many technical issues lead most researchers to purchase commercial autopilot systems[1] so they can focus on using the aircraft rather than integrating its components and optimizing their interactions. A complementary approach to building and testing the UAV is to purchase a commercial simulation package; this solution simply trades money for time.[2] The drawback to purchasing a commercial autopilot is the inability to customize the control system, which may be a fundamental limitation depending on the application of interest.[3,4] A detailed process for designing and testing UAV control systems is well known to experts in the area but has not always been well communicated, an oversight that often requires researchers to purchase a one-size-fits-all autopilot or reinvent the wheel in each new research effort.

To overcome the tradeoff between purchasing a black box autopilot and building one from scratch, this paper describes an approach to designing an autopilot consisting of the basic UAV components, including sensors, actuators and an airframe, to test several control system designs via a series of simulations. The resulting integrated UAV Control System Testbed can be used to explore the many types of control systems that have been developed for generic applications[5] without any significant hardware or software modifications. And while far more sophisticated UAVs and UAV simulation systems exist than will be presented here (e.g., Global Hawk and the X-45), the details of such aircraft are not widely available in the open literature. The simulations in these larger research projects are generally ill-suited for application to small-scale UAV efforts because of resource constraints.[6] On the other end of the spectrum are the relatively common UAV research efforts at universities; however, descriptions of those simulations tend to focus on high level architecture and are short on detail.[7-9]

The three main steps in this development of the hardware-in-the-loop (HITL) simulation and basic control laws are: 1) design and code the software drivers and other tools that interface the control system with the external, real world sensors and actuators; 2) design the simulations that will test the control system and implement that design in

* Aerospace Engineer, Automation Concepts Research Branch, M/S 210-10, AIAA Member

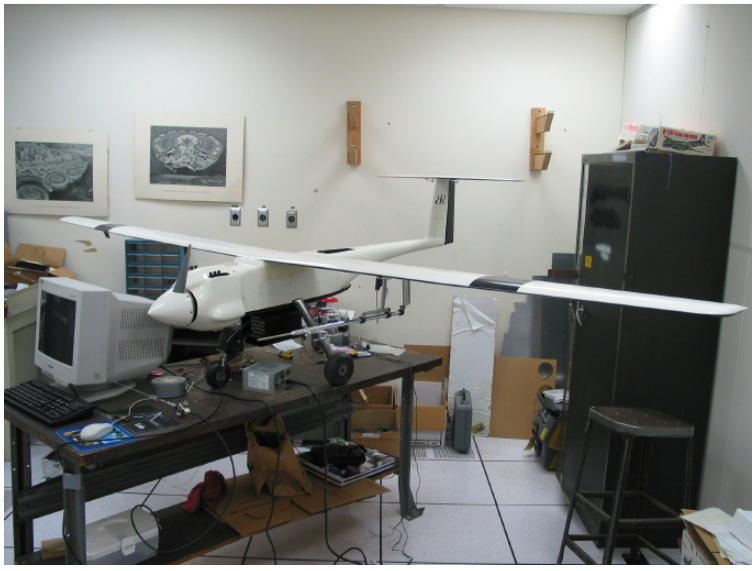

Figure 1. The UAV on its development bench.

a *Simulink*[†] simulation of the nonlinear dynamics, a *Simulink* simulation of the linear dynamics (which is used to develop the control system), and the final simulation that runs directly on the avionics computer; and 3) develop the control system and associated components, including the reference trajectory generator, sensor filters, and sensor models, and test them in the full nonlinear simulation based in the avionics computer. The next three sections describe each of these steps in detail.

II. Hardware-Software Integration

This section discusses the UAV's hardware and software systems generally and then focuses on the important hardware-software connections that must be used or emulated by the HITL simulation for high fidelity testing of the control systems.

A. UAV Systems

The primary components of the UAV for the simulation are the airframe, sensors, actuators, and avionics. The airframe and some of the sensors are displayed on the laboratory development bench in Figure 1. The airframe resembles a sailplane, a necessary feature for high endurance applications, has a 4-m wingspan, and is able to stay aloft on its one gallon tank for four hours. It was customized for UAV applications by the addition of a large payload pod on the underside of the fuselage (not shown in Figure 1) to accommodate the avionics computer and other research electronics. The avionics computer is a PC104 stack with a Pentium CPU, 256 MB of RAM, a 384 MB flash drive, an A/D conversion board, a radio modem board, and an integrated power conditioning unit with battery backup. A block diagram showing the electrical connections between the avionics, sensors and actuators is shown in Figure 9 in the Appendix. The operating system running on the avionics computer is QNX, a real-time operating system that guarantees a maximum execution time for each process. This capability ensures the critical command and control routines will always be executed on a rigorous schedule whether or not background processes are demanding CPU time.

The sensor suite on the UAV consists of an integrated IMU and GPS receiver, air-data probe, hall-effect switch measuring propeller shaft RPM, and hall-effect sensors mounted to each control surface servo to measure the deflection angle. The IMU contains three-axis accelerometers, magnetometers, and rate gyros (solid state), and is integrated with the WAAS-equipped GPS receiver to provide a filtered aircraft state vector. The air-data probe contains static and total pressure ports, and angle-of-attack and angle-of-sideslip vanes. It was custom mounted to the port wing of the UAV, and an electronic anti-aliasing filter was connected to the output of the differential pressure transducer to reduce high-frequency noise before A/D signal conversion. The hall-effect sensors are mounted to the servos and provide a useful check on their operation by giving advanced warning about the failure of any of the control surfaces to move to the desired deflection.

The flight control devices on the UAV are: throttle, ailerons, elevator, rudder, flaps and nose wheel direction. The actuators that drive these control surfaces are typical of any large radio-controlled aircraft: high speed, high torque hobby servos supplied by Airtronics. The servos take a Pulse-Width Modulated (PWM) input signal that commands an angular displacement, and an internal feedback mechanism maintains this displacement in the presence of disturbance inputs. The avionics compute the appropriate control surface deflections and send them all as a message packet over a serial line to an off-the-shelf servo controller. That controller interprets the commanded displacement and generates the appropriate PWM signal, which is the routed to the servo.

[†] Simulink is a commercial simulation package purchased from The MathWorks at www.mathworks.com.

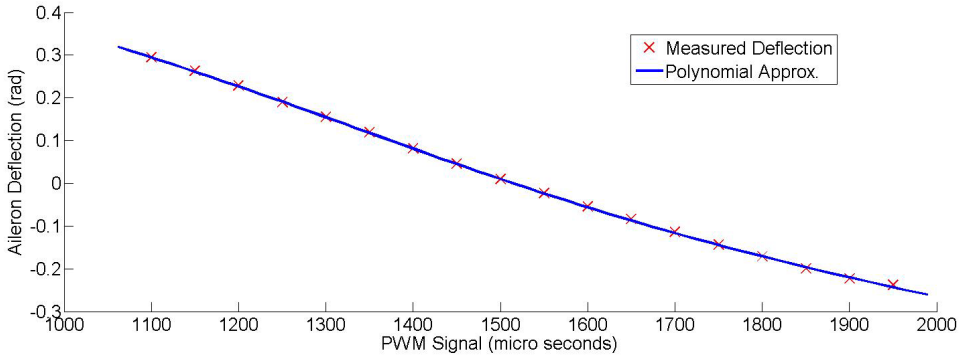

Figure 2. Measured aileron deflection and 3rd degree polynomial approximation for calibration.

B. Servo Control

A high fidelity simulation requires the avionics computer to send commands to the servos and read back servo deflections exactly as it would in flight. This incorporates into the simulation such realistic features as command and response delays and electronic noise effects. The avionics computer was physically wired to the servos via a serial connection to a commercial servo controller and a PWM signal to the servos. Functions were written to initialize the serial port, packetize the servo commands with a message header and checksum, and buffer the character strings that comprised the message so they could be written to the serial port. Special two-way command and control functions were also written to configure and communicate with the servo controller.

With the ability to send basic commands to the servos, the next task was to set direction conventions and place some software protections in front of any call to change the control surface deflections. The servo protection function was created by commanding each control surface to move to the limits of the physical linkage on the aircraft and measuring the value of the input at that point. The servo command function always checks the commanded deflection against this limit and changes the command to the limiting value if it is outside the predetermined bounds.

The PWM commands to the servos are in microseconds of high voltage (4.5 V, typically between 1 and 2 ms with an overall wavelength of 20 ms), so a conversion function between pulse width and radians was required. A table of the deflection values in radians versus pulse width was created for each actuator, and a third order polynomial fit to the data. An example of such a fit is given in Figure 2. This approach presumes the servo pulse widths will consistently drive the servos to the same deflections, and precludes changing the lever arms of the control surface or the orientation of the actuator. This assumption has been borne out over a year of operation as the servos consistently return to the same deflection for a given pulse width command.

C. Sensor Inputs

The physical connections between the sensors and the avionics computer come in two flavors: the integrated IMU/GPS connects over an RS-422 serial line, while the hall-effect sensors, differential pressure transducer, propeller shaft RPM sensor, and α and β angle sensors (angle of attack and angle of sideslip, respectively) connect to an A/D conversion board attached to the CPU.

A calibration of each sensor must be performed following completion of the physical wiring. The ideal approach, which provides calibration data for the RPM sensor, pressure transducer and angle sensors and helps refine the aircraft model, is to fly the UAV under direct pilot control from the ground using the RC link. This option leverages the software and hardware systems already onboard the UAV to measure and record experimental data, and saves a significant amount of time if the aircraft is designed to be flown by a remote pilot. The drawback to this approach is the need for a full safety analysis and hardware checkout at an early stage in the UAV's development. This calibration method is currently being pursued but has not yet been carried to completion.

The hall-effect sensors are calibrated by commanding the servos to move to the predetermined deflection points (which were measured during construction of the PWM-to-radian mapping) and averaging several dozen readings on each sensor at each of those positions. As noted above, this does not necessarily return the absolute deflection of the control surfaces if the servo positions drift over time, but it will allow a warning flag to be raised if the commanded deflection is significantly different from that measured by the sensor. A calibration matrix is automatically constructed as part of the initialization routine at power-up, and from that point on used to translate sensor readings into actual angular displacements.

The final consideration in writing the sensor driver software is to ensure the desired rates of sensor readings is not faster than the CPU can handle while it is computing control commands, communicating with the ground, and carrying out the myriad other tasks associated with piloting the UAV. The appropriate sensor rates were determined by this CPU speed consideration, the desired frequency of the individual data types, and the capacity of the wiring itself (a particular concern for the large data volumes coming across the relatively slow serial link). The sensor data rates selected from these considerations are displayed in Table 1.

Table 1. Sensor data rates

Sensor	Data Rate (Hz)
IMU	50
ADP – Differential Pressure	10
ADP – α and β vanes	10
Hall sensors	10
Propeller Shaft RPM	10
GPS	10

III. Simulation

The key to ensuring a UAV is safe and its control system robust is a rigorous program of testing and simulation. This can usually be accomplished by executing several cycles of design, test, and simulation at the highest possible fidelity, which this section describes. The simulations use two frames of reference: an Earth-Centered, Earth-Fixed (ECEF) frame and a body-axis frame, both of which are defined in the usual way.[1,10] The system used to reference Earth's geoid is WGS84,[10] and standard Euler angle conventions are used. When unit vectors are employed in the following sections the appropriate frame will be designated by using either $\boldsymbol{b}$ or $\boldsymbol{e}$ with the relevant subscripts (e.g., $\boldsymbol{e}_z$ for the Earth's rotation axis or $\boldsymbol{b}_y$ for the body-fixed pitch axis).

A. Nonlinear Simulink Simulation

The initial simulation of the UAV was implemented using the Mathworks' block diagram-based simulation utility, *Simulink*. This software was selected because design and construction of the simulation would be fairly rapid, a large number of add-on tools ("toolboxes") were available to simplify analysis of the models and control system, and because it is easy to expand and modify.

The heart of the non-linear simulation is a set of lookup tables that interpolate the values of the stability derivatives as functions of angle of attack and sideslip. These derivatives are converted to the appropriate coefficients (lift, drag, x-moment, etc.) through multiplication by some combination of aircraft and environmental states (dynamic pressure, reference lengths and areas, etc.) and summing of the various components of the specific coefficients (e.g., $C_L = C_{L0}+C_{L\alpha}+C_{Lq}+C_{Lailerons}+\ldots$). The derivative lookup tables themselves were obtained from a panel code developed at NASA Ames called *LinAir*.[11]

The highest level block diagram of the simulation is shown in Figure 3. The block labeled "UAV Model" contains all the necessary simulation environment data, including atmospheric conditions, actuator saturation, and aircraft dynamics. The "Sensor Model" block contains the sensor emulation code with baseline models of each sensor, noisy outputs of the actual environment variables to replicate sensor outputs, and unit conventions that match the actual sensors.

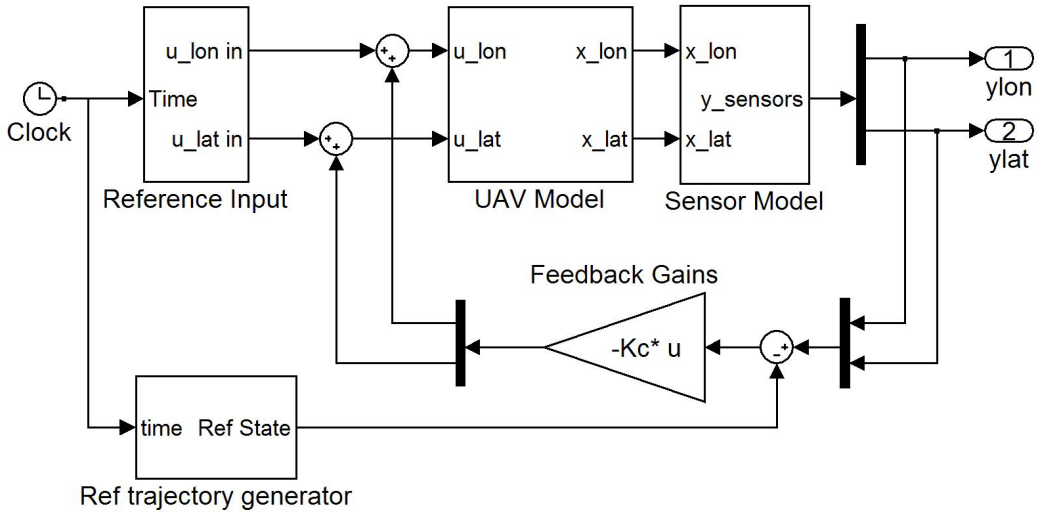

Figure 3. Block diagram of Simulink Simulation. Thick vertical bars represent the combination or separation of the longitudinal and lateral states.

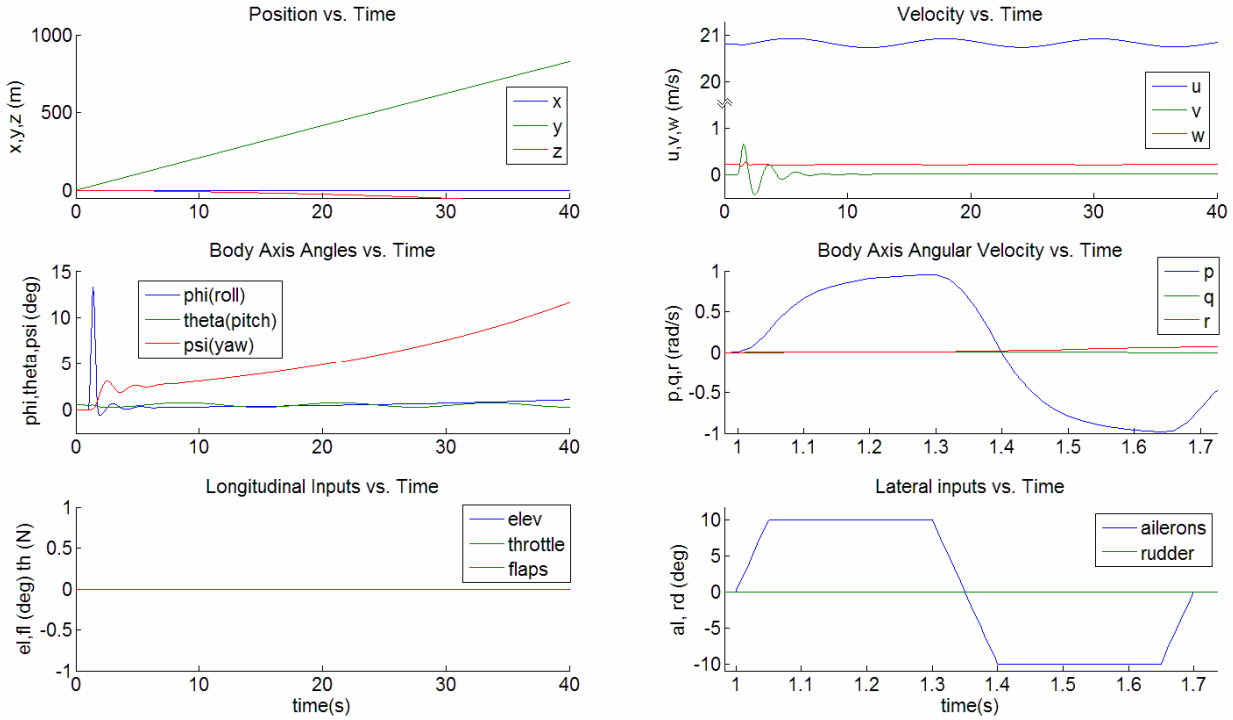

Figure 4. Response of UAV in Simulink non-linear simulation to a doublet input in ailerons.

The "Reference Input" block is primarily used in the validation of the simulation and is not the reference input that would normally be used to drive the aircraft along the nominal trajectory. The inputs required to follow the nominal trajectory are part of the simulation but are not shown on the diagram because they would unnecessarily clutter the figure. This block contains a series of switches that allow the experimenter to send a unit impulse, step, doublet, or zero input to any or all of the control surfaces. It is not intended for use with the feedback controller. This setup was found to be useful at each stage of the simulation because it built in a method for ensuring consistency at every step.

The simulation has several features beyond the standard aircraft dynamic equations that introduce greater realism to the test. Those dynamic equations are standard and can be found in many aerodynamics texts.[12,13] Features currently implemented in the simulation include a static wind field and atmospheric properties that are functions of altitude. Future work will implement a spatially complex, time-dependent wind field, a more detailed engine model, variation in the CG location, and other characteristics that improve the simulation's realism.

The response of the non-linear UAV model to initial conditions near the calculated equilibrium, assuming a nominal forward thrust, was checked to ensure satisfactory performance of the nonlinear simulation. The aircraft starts at an initial angle of attack of 0.61 degrees at a body-axis longitudinal velocity of 20.8 m/s, and remains at this equilibrium subject to the caveats discussed in the following section. The phugoid mode of the aircraft is approximately 10 seconds long, a reasonable number for an aircraft at this trim speed.

The response to the same initial conditions, but subject to a doublet input in ailerons is displayed in Figure 4. The longitudinal dynamics are little affected by this input, but the roll and yaw angles slowly begin to diverge. This yaw instability may be caused by a combination of the large payload pod centered on the fuselage and the small vertical tail. In any case, the slow divergence of the response suggests that mode should be controllable.

The next test of the simulation was with a step input to the elevator, the results of which are shown in Figure 5. The 3° input corresponds to a pitch-up maneuver, in response to which the aircraft begins to climb. The oscillations trade kinetic and potential energy with a period of about ten seconds, closely matching the phugoid mode.

Step, impulse, doublet and combinations of those inputs were tested for each actuator. The responses all show appropriate and expected behavior, lending credence to the conclusion that the non-linear *Simulink* simulation is accurate enough to design the control system and validate the stability of the system near equilibrium.

B. Linear Simulink Simulation

The next simulation step leading to the development of the control system is to linearize the equations of motion and check that the linear system matches the non-linear one for sufficiently large departures from equilibrium. This linearization can be done in several ways, all based on analyzing the contributions of the stability derivatives to the aircraft dynamics near a selected equilibrium value. While it is straightforward to come up with the state equations manually, *Simulink* provides a built-in linearization function that is fast and efficient and offers either manual or automatic selection of the operating point. The non-linear simulation was rearranged so that each control system input, given in Table 2, is represented by an input port, and each state of the control system an output port. This allows *Simulink* to use basic perturbation methods to calculate the effects of the inputs on the states near the selected equilibrium value. The operating point used for the control system design is the trimmed flight state, which was given in the above section.

Table 2. States and inputs of the linear UAV model

States: $\vec{x}$	**Inputs:** $\vec{u}$
x_e position (vertical)	δ_a ailerons
y_e position (longitudinal)	δ_e elevator
z_e position (lateral)	δ_r rudder
u_b long. body-axis CG velocity	δ_f flaps
v_b lat. body-axis CG velocity	δ_{th} throttle
w_b vert. body-axis CG velocity	
p aircraft angular rate about x_b	
q aircraft angular rate about y_b	
r aircraft angular rate about z_b	
φ Euler angle (roll)	
θ Euler angle (pitch)	
ψ Euler angle (yaw)	

Simulink will automatically create a state space representation of the system once it is given these parameters. This representation can be analyzed using Matlab's *step()* and *impulse()* commands; however, it is simpler and more consistent to place the state space model in a second *Simulink* simulation so that initialization and analysis routines written for the non-linear model can be reused. The matrices comprising the linear model may be found in the Appendix in Figure 10.

The response of the linear system showed good agreement with the non-linear system for small departures from equilibrium, and differences between the linear and nonlinear simulations are minor. The longitudinal velocity reaches a lower minimum in the linear simulation of elevator response than the non-linear simulation, and the pitch angle reaches a slightly higher maximum. The aileron doublet response is nearly identical in the two cases. This result suggests that a stable controller may be designed for the actual UAV using any of several methods based on the linear model, including LQR or LQG, successive loop closure, or a gain scheduled controller.

Linearization of the UAV dynamics allows a quick check of the validity of the UAV model with respect to the yaw instability shown in Figure 4. If the aircraft is truly unstable that fact should be reflected in the state transition matrix's eigenvalues as long as the linearization was carried out correctly. The state matrix has three unstable

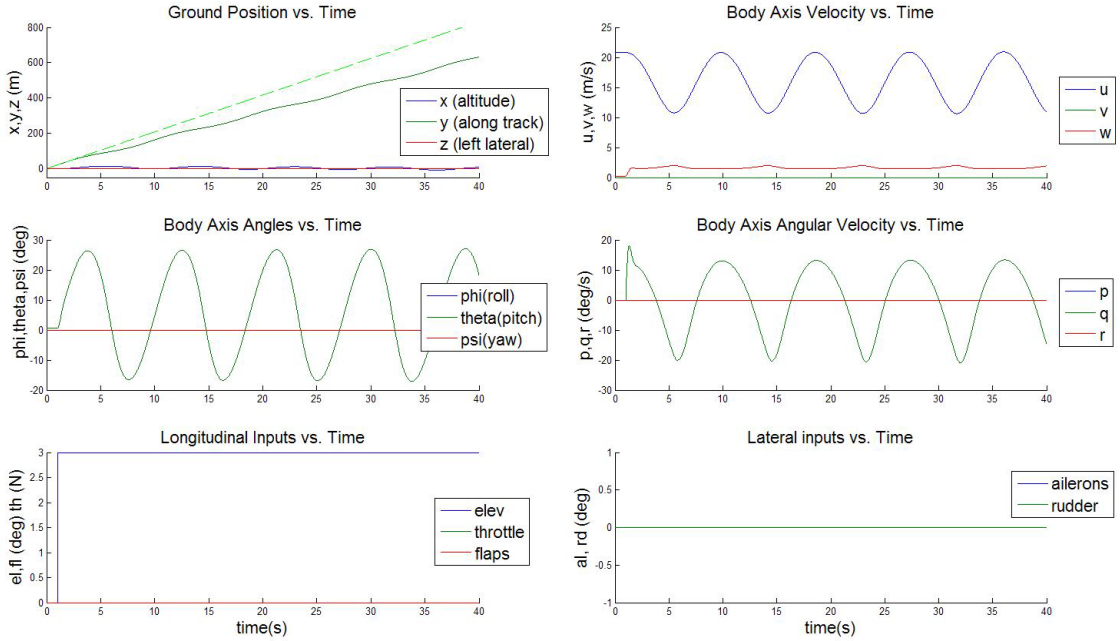

Figure 5. Response of UAV in Simulink non-linear simulation to a step input in elevator. Dotted green line in upper left graph is the unperturbed trajectory of the aircraft.

eigenvalues at x=0.02±0.66*i* and 0.08. A modal analysis shows that the conjugate pair of eigenvalues corresponds to an unstable phugoid mode, which causes the UAV to experience oscillations that grow to an amplitude of 10 m within about three minutes. The real eigenvalue corresponds to the spiral mode discussed in the above section and referenced as yaw instability. That instability is also slow; the linear model analysis indicates the yaw would reach 45˚ only after three minutes. The UAV has been flown manually by an RC pilot so these instabilities are slow enough not to require special compensation; however, the presence of such dynamics probably suggests the aerodynamic model should be revisited to ensure no other unforeseen problems are present.

C. Hardware-in-the-loop Simulation

Several approaches to implementing a hardware-in-the-loop simulation are possible. The first simulation architecture explored involved sending simulated IMU, GPS and air data probe data to the UAV's avionics from *Simulink* over a serial line. The hall-effect sensors could directly measure the servo deflections and so did not have to be simulated in this way. The avionics would then calculate the control commands based on the reference trajectory and real and simulated sensor data, and the control surface deflections would be sent back to *Simulink*. The onboard software would therefore be identical to the flight software except for the simulated sensor data. *Simulink* would handle the modeling of the atmosphere, environment and motions of the aircraft, thereby providing the highest fidelity testing of the autopilot possible without actually flying the UAV.

Unfortunately, this plan did not work because of the lack of an external interface (e.g., serial) from within the simulation and because the available version of *Simulink* did not allow real-time execution. This last problem was the critical limitation because the two simulations could not be synchronized without additional Mathworks toolboxes (*Realtime Workshop*). This approach is only mentioned as a lesson-learned in the hope that other researchers will not also attempt to apply commercial tools in ways they are not intended.

An alternative approach was explored in which the entire simulation was coded in a *Matlab* ".m" file. This alleviated the interface problem because a custom serial driver may be programmed in *Matlab*. It also partially solved the real-time execution issue by allowing time stamps to be exchanged at each integration step, which synchronized the two programs' execution. However, the execution of the simulation using this method was too slow for real-time application.

The most straightforward and flexible, if also most time consuming resolution to these problems was to code the entire simulation in *C* on the avionics computer. There would be no interface problems because all aspects of the simulation would run on a single machine, the program would execute quickly enough because *C* is a compiled rather than interpreted language, and the simulation could be executed in real-time by using blocking processes and wakeup timers. It is also an inexpensive option because no additional software is needed. The main drawbacks were the need to write a new, complex simulation program in *C* and to validate this simulation's operation against the *Simulink* model. However, some functions required in the final flight control software were written during the avionics computer implementation of the HITL simulation, so this approach did have ancillary benefits.

The state data received from the sensors during flight will contain non-negligible noise components, so the sensor data in the HITL simulation required comparable degradation. A relatively simple noise model was created for each sensor by measuring the mean and variance of the sensor output for a single operating point, usually when the sensor should be reporting a zero measurement. For instance, the noise was measured on the differential pressure transducer, which outputs the difference between total and static pressure, when the difference was theoretically zero. This method does ignore the potentially different error characteristics of the sensor when it is at the limits of its output (e.g., maximum pressure difference), but the method is a good first order approximation to the true flight environment and can be made more sophisticated with measurements taken during RC flight. A detailed discussion of the hall sensor noise model is given in Section IV-C.

Validation of the HITL simulation was done in the same manner as validation of the linear simulation: the response of the UAV to open loop impulse, step and doublet commands was measured and compared with the non-linear model in the *Simulink* simulation. In general, the HITL simulation shows good agreement with the non-linear simulation, but several minor differences remain despite extensive debugging and laborious comparisons. An example of the UAV's operation in the HITL simulation, subject to control inputs, will be discussed in Section V. Most of the state differences between the two simulations for a given control input are small (less than 10%), and the benchmark comparison cases (elevator step and aileron doublet) accurately reproduce the results shown in Figure 4 and Figure 5.

IV. Control System

The HITL simulation's primary purpose is to assess the control system's stability, error characteristics, and speed of response. This section discusses several remaining aspects of the overall control system, including creation of a reference trajectory, modeling the servos for inclusion in the overall control system, designing a filter for the hall effect sensors to enable closed loop feedback of servo positions, and the design of a basic controller.

A. Reference Trajectory

The generation of a reference trajectory proved to be one of the most interesting and challenging aspects of the project. In general, the existence of a reference trajectory is frequently taken for granted in control system design. However, a number of informative texts on the subject do exist that were used as a starting point for the current design.[14-18] For this UAV application, the goal was accurate tracking of position with time. A two-stage process was designed to provide the reference trajectory: the first would be a hybrid analytical/heuristic algorithm to place a continuous trajectory between each waypoint, assign a velocity to segments of the trajectory, and rebuild an auxiliary path back to the original trajectory when the UAV passes a given error threshold; the second stage creates the full-state reference, providing each state listed in Table 2 at any time along the trajectory.

The approach to designing a control system with which to exercise the HITL simulation was to use a simple Linear Quadratic Regulator (LQR) because this method would quickly provide adequate feedback gains. A drawback of the LQR, however, is the need to know the full reference state at every time step. While this is relatively easy to do for position, states like the Euler angles are not nearly as straightforward. The most general approach would be to numerically solve the full set of nonlinear equations that describe the aircraft's dynamics; however, a simpler approach is to calculate a rough approximation to the less important states and design the LQR so that errors in those states carry little cost. Only position or velocity state errors would be heavily weighted, so potentially good trajectory following could occur even without good a priori knowledge of the complete reference state. The equations used in this method are:

$$\psi(t) = \tan^{-1}\left[\frac{z_e(t) - z_e(t-1)}{y_e(t) - y_e(t-1)}\right]$$

$$\theta(t) = \gamma(t) + \alpha(t) \cong k * \sin^{-1}\left(\frac{-\dot{x}_e(t)}{u_b(t)}\right) \qquad (1)$$

$$\text{with } k \approx 2 \text{ and } \gamma \text{ the vertical flight path angle}$$

$$\phi(t) = \tan^{-1}\left[\dot{\psi}(t)\frac{v_b(t)}{g}\right]$$

The Euler angles and ground position variables can be numerically differentiated and filtered, and the remaining state variables (body axis velocities and body axis angular rates) calculated from the aircraft dynamical equations.[12] The poorest assumption made in Eq. (1) was to set the constant in the θ equation equal to 2. This set the angle of attack for a zero climb rate equal to the current flight path angle (since the inverse sine term is the flight path angle). For the expected slow climb rates of this aircraft that approximation should be sufficient. Some simultaneous solution or iteration of the above equations is necessary, but the problem has been significantly simplified while remaining accurate enough to permit a control solution, as will be shown in Section V.

B. Servo Model

The servo model was computed by feeding a chirp signal into the servos and measuring the response using the hall sensors. Because the hall sensors have a direct analog connection to the avionics computer and that signal is being sampled at 200 kHz (roughly 10 kHz per hall sensor) it was assumed that the sensor output would be sufficiently fast when compared with the mechanical servo response that its dynamics could be neglected. The input signal began at a frequency of π rad/s and increased linearly with time to 5π rad/s 30 seconds later, with an amplitude equal to the maximum mechanical range of the servo. A bode plot of the servo response computed using the *Matlab SystemID* Toolbox is shown in Figure 6.

Several servo models of varying order were created by feeding this data into the *Matlab SystemID* Toolbox, with the transfer function of the best fitting model being:

$$\frac{Y(s)}{U(s)} = \frac{216}{(s+14.7)^2} \tag{2}$$

To validate the servo model a set of full range of motion step inputs and constant amplitude, high frequency inputs was sent to both the servos and the servo model. It was observed that the model predicts the output of the servos well. The model was then inserted into the *Simulink* simulation, linearized around zero degrees of deflection to obtain the linear model, and coded into the HITL simulation. The above simulation results incorporated only simple rate and saturation limits, but the following control system tests use the model given in Eq. (2).

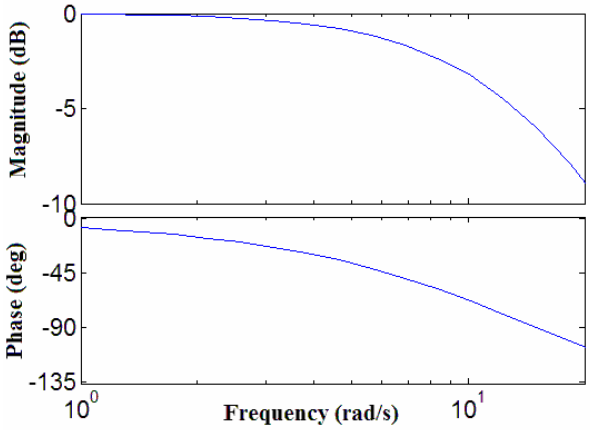

Figure 6. Bode plot of the servo response, obtained from an input chirp signal between π and 5π rad/s.

C. Kalman Filter for Control Surface Sensing

It was necessary to measure the accuracy of the hall sensors to determine whether they could be used for closed-loop measurement of the control surface deflections, or whether the servo output had to be modeled from the open-loop servo commands. The manufacturer of these particular hall-effect sensors describes the output (over 60 degrees) as ranging between 0.5 and 4.5 V, with an accuracy of 0.025 V. The accuracy over such a range is just 0.375 degrees, which is about 10% of the full scale magnitude. To confirm these data and quantify the magnitude of the sensor noise the output of the hall sensor was measured when the servo was stationary. The raw output of the hall sensor, mounted to the flap servo in this case, is shown in Figure 7 in blue. The mean value of the output noise is -0.0639 degrees, the variance is 0.055 degrees, and the range is 1.045 degrees.

A Kalman filter can often reduce the random noise from a sensor and provide a better estimate of the actual output. The measured error distribution of the hall sensor is not perfectly Gaussian, but it is close enough to consider a Kalman filter implementation. The canonical filter assumes only a white noise input, but in this case the servo is being driven by the control system and so the input is not entirely random. However, the system in question is linear so the estimate can be decomposed into deterministic (servo commands) and non-deterministic (hall sensor noise) components and each estimated separately.[19] The deterministic estimate is trivial since it is just the output of the above servo model, while the non-deterministic noise input is estimated from the noise mean and variance values. The improved output using this technique with a zero input signal, along with the raw signal, is shown in Figure 7.

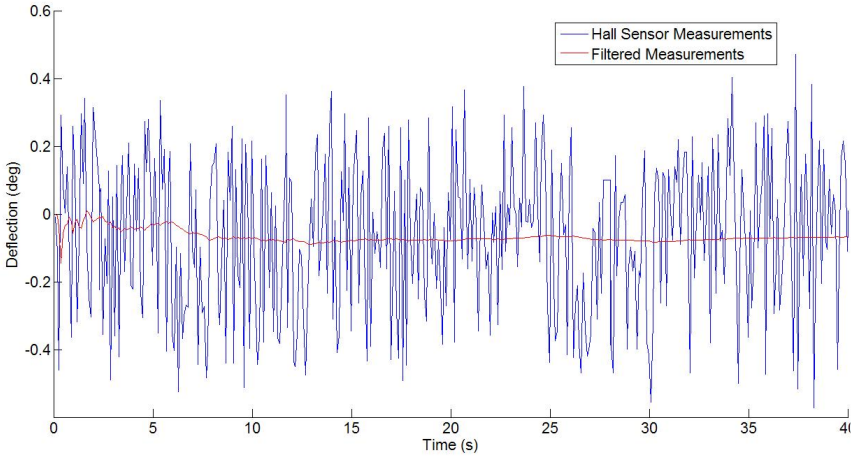

Figure 7. Kalman filter results on hall sensor output given no input to the servos.

D. Control Law Development

The most successful control strategy tested on the UAV was an LQR operating on the 12 states listed in Table 2, with the success criterion being the minimum position error with respect to the reference trajectory. Other strategies that were examined but never carried to an equivalent level

of maturity included successive loop closure, feedback linearization, model predictive control and generalized predictive control.[5] In the LQR strategy the continuous time state space representation of the linearized model was converted into a discrete time representation according to:

$$\begin{aligned} A_d &= \sum_i^{\infty} \frac{1}{i!} A^i dT^i \\ B_d &= A^{\perp}(A_d - I)B \\ C_d &= C \quad D_d = D \end{aligned} \tag{3}$$

where the subscript ***d*** represents the discrete equivalent, $A^{\perp}$ is the pseudo-inverse of ***A*** (because ***A*** is not of full rank $\boldsymbol{A}^{-1}$ does not exist), and ***dT*** is the time step of the discrete system. The weighting matrices ***Q*** and ***R*** of standard LQR design, which determine the cost of non-zero values of the state error and input, respectively, were selected so that errors in position cost significantly more than errors in the other states. The flap inputs are penalized more than the other inputs because the use of flaps outside takeoff and landing operations is not a standard way of controlling an aircraft. A parameter $\boldsymbol{\mu}$ multiplies ***Q*** to change the relative cost of state errors versus inputs. For this first, basic implementation of the control system the approach was to simply increase the value of $\boldsymbol{\mu}$ until the UAV response to position errors greater than 20 m was unstable. When errors greater than that threshold occur the avionics replan the reference trajectory.

V. Control System Results in Simulation

The following results of the control system design were obtained using the full state LQR described in the above section, and only the position error between the reference and actual trajectories was used to calculate the control inputs. For the *Simulink* non-linear simulation the controller is continuous time, and the tracking of a sinusoidal reference trajectory is relatively good with position errors less than 1 m. Control power requirements were not excessive, with a maximum aileron deflection of about 3˚.

The HITL simulation was run under the same conditions as the *Simulink* non-linear simulation: no nominal, feed forward inputs to anticipate turns, a servo model to provide rate and deflection limiting, and artificial noise added to

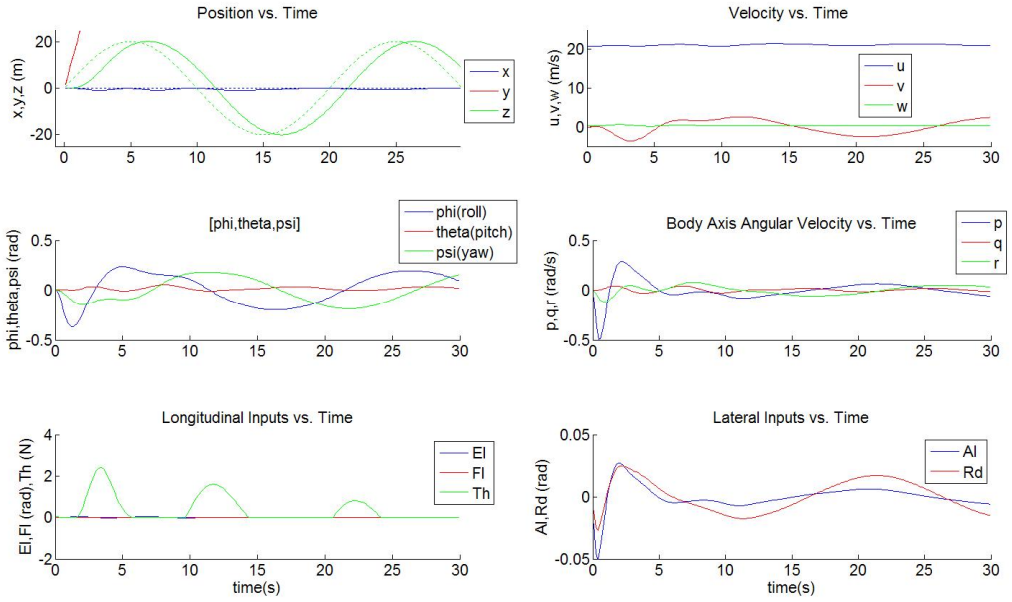

Figure 8. UAV closed loop response in HITL simulation, using an LQR to track a sinusoidal input in the lateral direction: zref=20*sin(2πt/20).

the sensor outputs. Noise was also present in the hall effect sensors because they were not being simulated. The flight control software had been written and used in the simulation to generate a reference trajectory, measure and record the sensor outputs, and model the aerodynamics of the UAV.

The response of the UAV in the HITL simulation was measured for a sinusoidal lateral reference trajectory of amplitude 20 m and period of 20 seconds, and is shown in Figure 8. The tracking of this reference input is not as accurate as it is in the *Simulink* simulation, a result likely due to the additional noise in the hardware systems. Better tracking can be obtained with a higher value of μ (the tradeoff between input cost and state error cost), but the full control system will use the nominal feed forward inputs to track a sinusoidal reference trajectory and so the higher gain may be counter productive, especially in the presence of increased sensor noise.

VI. Conclusions and Future Work

This paper has discussed the integration of an avionics computer with sensors and actuators, designing and implementing several types of simulations to assist in the design, implementation, and testing of a control system, and evaluating a control law through an HITL simulation. The simulation's implementation has largely been completed, although additional functions remain to be added to increase overall fidelity. Examples of near-term changes include a better model of the propulsion system, addition of more realistic sensor noise characteristics including biases, and inclusion of the nominal inputs required to follow an arbitrary trajectory.

A control system has been designed, coded and tested, and shows decent dynamic response characteristics for a first implementation, but significant additional work will be required to field a flight-ready system. The common approach of linearizing the dynamics around several operating points and mode switching between them during flight will likely be discarded in favor of more general techniques, including feedback linearization and generalized predictive control.

A short-term goal of the project is to evaluate additional simulation architectures using more sophisticated analysis tools to automate the process of simulation, control system design, and coding. Tools that warrant such evaluation are the *Realtime Workshop* (*RTW*), *RTW Embedded Coder* (which writes *C* code for *RTW*), the *Stateflow* and *Stateflow Coder* toolboxes, and the *Aerospace Blockset*. These Toolboxes will allow the construction of a more sophisticated and flexible simulation, design of the control system within *Simulink*, and automatic generation of the code required for the flight control system. The tradeoff in their use is the additional time spent learning how to apply them to a UAV project and their considerable cost versus the higher anticipated productivity and increased UAV functionality and reliability.

Appendix

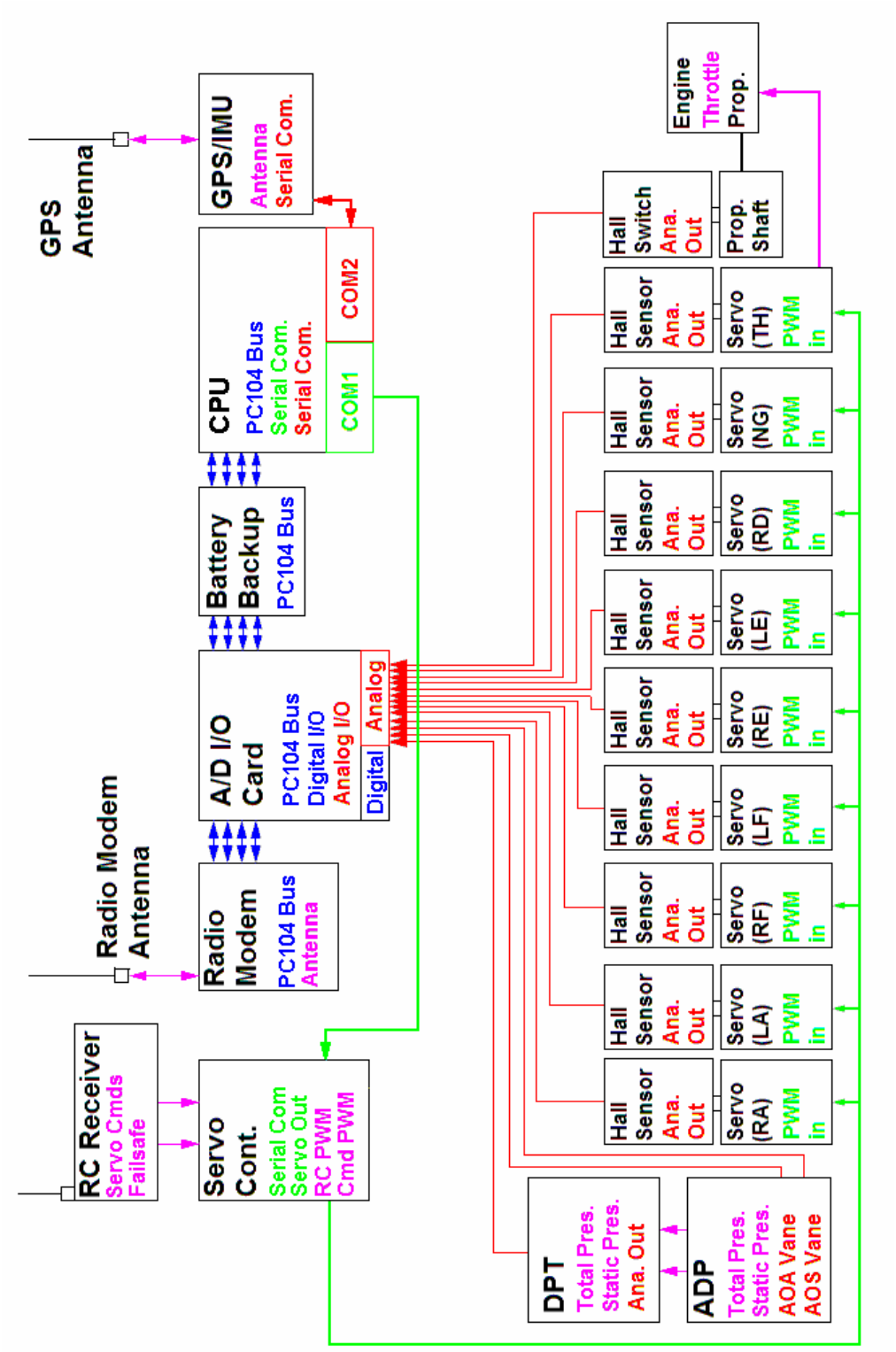

Figure 9. Block diagram of UAV hardware systems. DPT is the differential pressure transducer, green lines are commands coming from the avionics, red lines are sensor data, magenta lines are analog signals, and blue lines digital signals. The name of each component is at the top of the box in black, followed by the names of the inputs and outputs color coded according to signal type.

$$\mathbf{A}\bar{\mathbf{x}}=\begin{bmatrix} -0.0064 & 0 & 0.3271 & 0 & -0.1121 & 0 \\ 0 & -0.4588 & 0 & 0.2202 & 0 & -20.4883 \\ -0.7534 & 0 & -3.8583 & 0 & 20.2285 & 0 \\ 0 & -1.0897 & 0 & -15.3509 & 0 & 3.7213 \\ 0.2211 & 0 & -0.9395 & 0 & -4.2176 & 0 \\ 0 & 0.9490 & 0 & 0 & 0 & -0.9796 \end{bmatrix}\begin{bmatrix} u_b \\ v_b \\ w_b \\ p \\ q \\ r \end{bmatrix}$$

$$\mathbf{B}\bar{\mathbf{u}}=\begin{bmatrix} 0 & -0.0015 & 0 & -0.0037 & 0.0517 \\ 0 & 0 & -0.0450 & 0 & 0 \\ 0 & -0.1046 & 0 & -0.3618 & 0 \\ -1.6247 & 0 & 0 & 0 & 0 \\ 0 & -0.8175 & 0 & 0.1973 & 0 \\ 0 & 0 & 0.1900 & 0 & 0 \end{bmatrix}\begin{bmatrix} \delta a \\ \delta e \\ \delta r \\ \delta f \\ \delta th \end{bmatrix}$$

$$\mathbf{C}=\mathbf{I}_{6\times 6} \qquad \mathbf{D}=[0]$$

Figure 10. Reduced-order state space matrices of the UAV model linearized about operating point given in Section III. The states and inputs are defined in Table 2. Position and angular states are not shown.

Acknowledgments

E. Mueller thanks Matt Jardin, his colleague on the early work described in this paper, for initiating the project and providing overall management direction along with significant technical contributions. He would also like to thank current and former management in the Aviation Systems Division for providing the financial support and freedom to pursue this project, particularly Tom Edwards and Banavar Sridhar. Finally, thanks go to the reviewers of this paper for their hard work to improve its content: Gano Chatterji and Russ Paielli.

References

[1] Jodeh, N., Blue, P., Waldron, A., "Development of Small Unmanned Aerial Vehicle Research Platform: Modeling and Simulating with Flight Test Validation," *AIAA Modeling and Simulation Technologies Conference*, Vol. 1, AIAA, Washington, DC, 2006.

[2] Christhilf, D., Bacon, B., "Simulink-Based Simulation Architecture for Evaluating Controls for Aerospace Vehicles (SAREC-ASV)," *AIAA Modeling and Simulation Technologies Conference*, Vol. 1, AIAA, Washington, DC, 2006.

[3] Cox, T., Nagy, C., Skoog, M., Somers, I., "Civil UAV Capability Assessment," NASA Dryden Research Center, Edwards, CA, December, 2004 (unpublished).

[4] Harris, Z., "Survey of UAV Applications in Civil Markets," *Proceedings of the 9th Mediterranean Conference on Control and Automation*, Dubrovnik, Croatia, June, 2001.

[5] Soloway, D., and Haley, P., "Neural Generalized Predictive Control," *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, Dearborn, MI, September 1996.

[6] Lux-Baumann, J., Dees, R., Fratello, D., "Control Room Training for the Hyper-X Program Utilizing Aircraft Simulation," *AIAA Modeling and Simulation Technologies Conference*, Vol. 1, AIAA, Washington, DC, 2006.

[7] Johnson, E., Mishra, S., "Flight Simulation for the Development of an Experimental UAV," *AIAA Modeling and Simulation Technologies Conference*, Vol. 1, AIAA, Washington, DC, 2002.

[8] Jang, J. and Tomlin, C., "Autopilot Design for the Stanford DragonFly UAV: Validation through Hardware-in-the-Loop Simulation," *AIAA Guidance, Navigation and Control Conference*, AIAA, Washington, DC, 2001.

[9] Goktogan, A., Nettleton, E., Ridley, M., Sukkarieh, S., "Real Time Multi-UAV Simulator," *IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, 2003,

[10] Misra, P., and Enge, P., *Global Positioning System: Signals, Measurements and Performance*, 2nd ed., Ganga-Jamuna Press, Lincoln, MA, 2006, Ch. 4.

[11] Durston, D.A., "LinAir: A Multi-Element Discrete Vortex Weissinger Aerodynamic Prediction Method," NASA TM 108786, 1993.
[12] Bryson, A., *Control of Spacecraft and Aircraft.* Princeton University Press, Princeton, NJ, 1994.
[3] McRuer, D., Graham, D., and Ashkenas, I., *Aircraft Dynamics and Automatic Control*, Princeton University Press, Princeton, NJ, 1973.
[14] Kaminer, et. al., "Trajectory Tracking for Autonomous Vehicles: An Integrated Approach to Guidance and Control," *AIAA Journal of Guidance, Control and Dynamics*, Vol. 21, No. 1, Jan-Feb 1998, pp 29-38.
[15] Fryxell, D. et. al., "Navigation, Guidance and Control of AUVs: An Application to the Marius Vehicle," *Control Engineering Practice*, Vol. 4, No. 3, 1996, pp. 401-409.
[16] Trankle, T, and Bryson, A., "Control Logic to Track Outputs of a Command Generator," *AIAA Journal of Guidance and Control*, Vol. 1, No. 2, 1978, pp. 130-135.
[17] Verma, A., and Junkins, J., "Inverse Dynamics Approach for Real-time Determination of Feasible Aircraft Reference Trajectories," *Atmospheric Flight Mechanics Conference*, Vol. 1, AIAA, Washington, DC, 1999.
[18] Gorder, P., "Trajectory Synthesis and Tracking as a Means of Flight Control Decoupling," *22nd IASTED International Conference on Modeling, Identification and Control*, Innsbruck, Austria, 2003, pp. 19-24.
[19] Brown, G., and Hwang, P. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, Canada, 1992, pp 287-288.