AIAA Infotech@Aerospace Conference
and
AIAA Unmanned...Unlimited Conference
6 - 9 April 2009, Seattle, Washington

AIAA 2009-1843

Simulink Based Hardware-in-the-Loop Simulator for Rapid Prototyping of UAV Control Algorithms

Mariano I. Lizarraga*, Vladimir Dobrokhodov†, Gabriel H. Elkaim‡,
Renwick Curry§, Isaac Kaminer¶

This paper describes a recently developed architecture for a Hardware-in-the-Loop simulator for Unmanned Aerial Vehicles. The principal idea is to use the advanced modeling capabilities of Simulink rather than hard-coded software as the flight dynamics simulating engine. By harnessing Simulink's ability to precisely model virtually any dynamical system or phenomena this newly developed simulator facilitates the development, validation and verification steps of flight control algorithms. Although the presented architecture is used in conjunction with a particular commercial autopilot, the same approach can be easily implemented on a flight platform with a different autopilot. The paper shows the implementation of the flight modeling simulation component in Simulink supported with an interfacing software to a commercial autopilot. This offers the academic community numerous advantages for hardware-in-the-loop simulation of flight dynamics and control tasks. The developed setup has been rigorously tested under a wide variety of conditions. Results from hardware-in-the-loop and real flight tests are presented and compared to validate its adequacy and assess its usefulness as a rapid prototyping tool.

I. Introduction

Hardware-in-the-loop (HIL) simulators are essential tools for flight control systems development. In essence, a HIL simulator generates synthetic data as if it was from the onboard sensors. This data is sent to the actual flight avionics which in turn, produces control commands. These control commands are relayed back to the simulator thus closing the control loop.[1] Specifically, UAV HIL simulators are generally comprised of: **(i)** the avionics, **(ii)** software to simulate the aircraft dynamics, engine, weather, actuator dynamics, etc. which, in further discussions will be referred as *the plant model*; and **(iii)** the hardware and software that allows *the plant model* simulator to send synthetic data to the avionics and receive control commands back.

These simulators have long been used for rapid prototyping of flight control systems in missiles,[2,3] helicopters[4] and fixed-wing aircraft.[5] Their wide adoption is mainly due to the fact that adequate simulation allows to save time and money when compared to real flight tests.[6] For instance, in missile development, it is estimated that the cost of a single firing test covers the cost of somewhere between 3,000 and 10,000 HIL simulated firings.[2] Another advantage of HIL simulators is that it allows preliminary experimentation of high-risk scenarios such as fault-tolerance[6,7] without actually exposing the hardware. This is extremely important, especially at the initial phases of the control design process. Another advantage, provided that the software simulating *the plant model* is accurate, is that the amount of parameter tuning when transitioning from the HIL to real flight implementation is greatly reduced.

There have been recent reports in the literature by several research laboratories employing HIL simulators for rapid prototyping of UAV flight control algorithms. Johnson and Schrage reported on Georgia Tech's rotary wing UAV research platform, the GTMax, which included a HIL simulator developed in-house written in C++.[8,9] Tisdale et al. reported on UC Bekeley's UAV platform which made use of the commercially

*PhD. Candidate, Baskin School of Engineering, UC Santa Cruz. Student member AIAA; malife@soe.ucsc.edu.
†Research Assistant Professor, Dept. of Mech. & Astronautical Eng., Naval Postgraduate School. Member AIAA; vldobr@nps.edu.
‡Assistant Professor, Baskin School of Engineering, UC Santa Cruz. Member AIAA; elkaim@soe.ucsc.edu
§Adjunct Professor, Baskin School of Engineering, UC Santa Cruz. Member AIAA; rcurry@ucsc.edu
¶Professor, Dept. of Mech. & Astronautical Eng., Naval Postgraduate School. Member AIAA; kaminer@nps.edu.

available Piccolo autopilot[10] and its HIL simulator to develop and flight test autonomous collaborative patrolling[11] . More recently Ref.[12] reported on the Naval Postgraduate School's Rapid Flight Test Prototyping System which made extensive use of the HIL simulator available with the Piccolo autopilot for development of vision-based target tracking, 3D path following, SUAV control over the network and high-resolution imagery on the fly.[12] Nevertheless, all of these HIL simulators share the same limitation, namely a lack of a simple and intuitive way to modify *the plant model*.

It is a fact that many – if not all – of the current Commercial Off-The-Shelf (COTS)[10,13,14] and open source[15] autopilots offer software simulators to perform HIL testing, and training. This software allows an end-user to conduct overall system check, training, and, in the Research and Development case, some limited algorithm verification and validation without the need to fly. Although the simulators shipped with currently available autopilots work well in a generic flight scenario, they are quite limited once a researcher tries to simulate complicated phenomena or use a different aircraft model. For example, simulating a component failure (control surface, sensor), a combination of them, or employing a different *plant model* from the one pre-programmed is difficult and often impossible.

This is why research labs involved in advancement of control algorithms require a more flexible and versatile HIL simulator. Ideally the HIL simulator should be an enabling tool that can easily be modified to adequately simulate the conditions under which a real flight will be conducted. The setup should be capable of being gradually extended from a simple linear *plant model* during the initial design phases, all the way to a rigorous and detailed non-linear one once/if the project so requires. Another desired feature is that the end-users should be able to add plotting and logging capabilities with ease. To the authors' knowledge there is currently no HIL simulator readily available that enables all of these features for the development of UAV control. Currently available HIL simulators do offer a limited capability to modify a few parameters of a fixed *plant model*; while the Open Source ones offer freedom to modify anything, there is an inherently steep learning curve requiring significant proficiency not only in control systems design but in a high level programming language. These are the key factors that contributed in deciding to develop a flexible and easily modifiable HIL simulator. The newly-developed HIL architecture has the ability to simulate any dynamic model using the inherently powerful capabilities of Simulink[16] modeling environment. Simulink-based design is then used to generate synthetic sensor data for a given UAV autopilot and has the capability to receive the control commands back. This offers a great flexibility not previously available.

From the previous discussion it is easy to infer that central to the HIL simulator is the mathematical description of *the plant model*. Undoubtedly the most crucial part of *the plant model* is that of the aircraft dynamics. Numerous works have been published solely addressing the issue of adequate modeling of aerodynamics and flight characteristics; ranging from a simple approximation using panel methods,[17] to a full regression model,[18] or a hybrid between the two.[19] The system identification process briefly presented in this paper is similar to that of Ref.[19] in the sense that it first also employs a panel method. However, for the final identification step that includes coupled interaction between the longitudinal and lateral channels it uses the less-known Parameter Space Investigation (PSI) method.[20] The PSI method is a rigorous tool that first proves an existence of a feasible solution in the multidimensional design variables space therefore explicitly delivering the corresponding feasibility bounds and then allows for a Pareto optimal result to be found.

The main contribution of this paper is the development of a new versatile HIL simulator architecture that greatly simplifies new UAV control system design. This new Simulink-based architecture allows for simulation of complex dynamic models and phenomena in one of the most versatile and convenient design environments supported by a wide variety of Control Design toolboxes. This reduces the need for high-level language programming and manual coding to simulate the environment where the control design will be tested. In turn, this will allow many other UAV research labs to test their algorithms for different classes of airplanes of any configuration under any flight conditions without risking their flight hardware. This capability extends the scope of their research by significantly reducing many man-hours of tedious implementation in a high-level programming language.

The outline of the paper is as follows: Section II presents in detail the HIL development, specifically the Simulink-to-autopilot communication and the architecture of the simulator, then it briefly presents the system identification method employed to obtain *the plant model*. Section III presents comparative results of the HIL simulation and the real-flight test; specifically the open-loop doubletand frequency responses are presented. Section IV presents a discussion on the computational performance of the architecture and ways of improving it. Finally concluding remarks provide a summary and a direction of future research.

II. HIL Simulator Development

The Piccolo Plus autopilot[10] is the core element of the current Rapid Flight Test Prototyping System (RFTPS)[12,21] developed at the Naval Postgraduate School (NPS). The principal idea of this system consists in integrating the flight control and prototyping computational platform onboard the airplane with the autopilot. This placement allows for efficient control development using xPC Target[22] tool as a development platform and eliminates most of the communication issues inherent to RF transmission. Although the following discussions pertain to the Piccolo communication interface version 1.3.2 and was developed with openly available documentation from the Cloud Cap Technology,[10,23] the core idea of this development can be easily applied to the latest versions of the Piccolo communication software as well as to any other autopilots currently on the market.

The Piccolo autopilot interacts with its own HIL simulator via a Controller Area Network (CAN) bus[24] using a well defined protocol. This communication protocol is noticeable simpler than the standard Piccolo Communications Protocol used in the normal autopilot-to-ground station communication. The autopilot supports CAN bus messaging to send control surface commands and receive simulated sensor readings from a PC via a CAN to USB converter unit.

In order to replace the standard Piccolo Simulator with a new Simulink-based one, three major tasks were undertaken: (**a**) develop an interface to the CAN/USB data bus and make that data available in Simulink; (**b**) develop and identify a six-degree-of-freedom (6-DOF) dynamic model of the plant including aircraft aerodynamics and engine model; and (**c**) establish the communication architecture under which the new HIL simulator would interact with the autopilot. The following Sections describe each of this components in detail.

A. Simulink-Autopilot Communications

The Piccolo autopilot communicates with its HIL simulator via a CAN 2.0 B bus using the extended (29 bits) frame identifier and an 8 byte data field which contains the actual payload (see Figure 1). The identifier itself is divided into three sections: the *Group* which classifies which type of message it is, the *Message ID* which uniquely identifies what data is being sent and the *Autopilot Address* to/from which the payload is sent. Overall, there are twelve different types of incoming messages (sensor data) and seven outgoing (control commands).

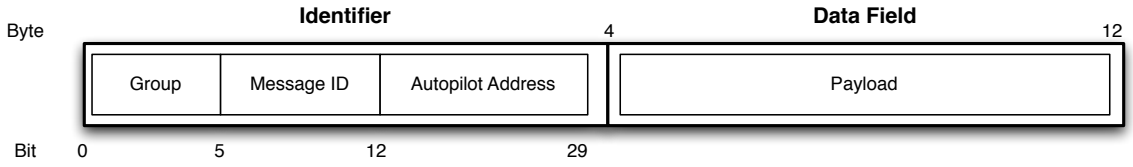


Figure 1: CAN message structure to communicate with the Piccolo autopilot

At the hardware level, Cloud Cap Technology provides a Systec USB-CAN Modul1 CAN to USB converter unit to enable communication of the autopilot with its own HIL simulator. Systec, as a part of the Windows operating system driver, also provides a thoroughly documented[25] Dynamically Linked Library (DLL) which can be accessed from any high-level programming language to read and send CAN messages.

Using this DLL, a C++ application was developed to act as a proxy that would receive commands from the autopilot and re-route them to Simulink. At the same time it receives sensor data form the Simulink model and re-routes it to the Autopilot. During the early design phases it was decided to use the User Datagram Protocol (UDP)[26] as the communications protocol between the newly developed application and Simulink. By using UDP to communicate with Simulink model, the application gained the additional advantage of being able to run on a different PC connected to the same computer network. During the later phases of the development, this proved to be a big advantage since the computer running *the plant model* required a significant amount of computing resources. Therefore, it was decided to share the modeling task between two different PCs.

B. Developing the 6-DOF Plant Model

As previously discussed, availability of an adequate *the plant model* is key to the success of any HIL Simulator. Although very important and time consuming by itself, the system identification part of the project will be briefly presented, providing enough details to understand the concept. Although the aerodynamics of a small UAV with standard high-wing configuration employed for this project is traditional and in general it is not difficult to identify, there are multiple steps involved in deriving a reliable model. The following strategy of system identification (see Figure 2) was employed:

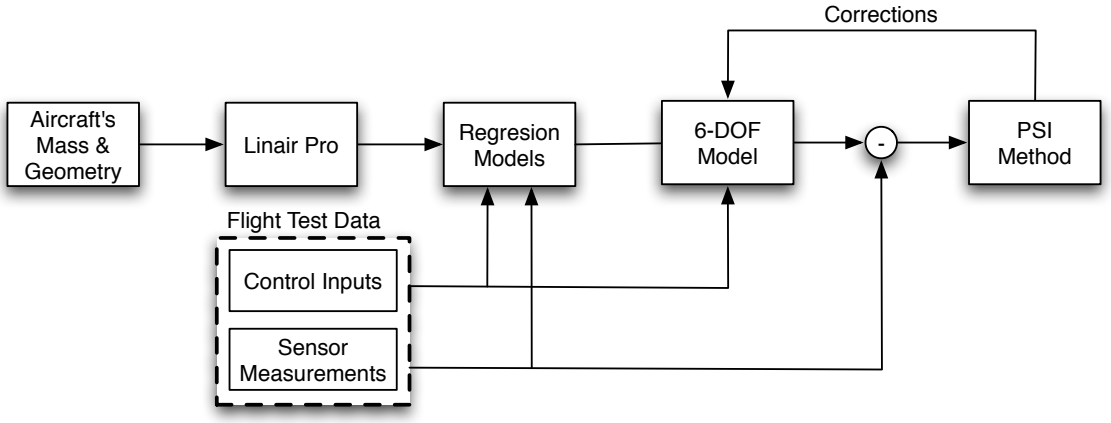


Figure 2: System identification architecture

Starting with initial measurements of the mass and geometry characteristics of the aircraft, the LinAir panel method software[27] was employed to produce initial estimates of basic aerodynamic terms. These terms were represented by the traditional regression model[18] for the aircraft configuration. Next a series of flight experiments were designed to measure airplane responses to a single-input (one channel at a time) doublet commands. Instrumentation of the airplane to measure data followed the general instrumentation recommendations found in Refs.[18,28] and references therein. To acquire the response data the built-in capability of Piccolo autopilot was used; it allowed for 100 Hz data sampling during the interval of the preprogrammed doublet. A series of doublets for open-loop system were performed separately in aileron, rudder and elevator channels in order to excite the UAV so that the data contained sufficiently rich information for accurate flight dynamics identification. The classical regression-based off-line identification techniques[18] were used to improve initial estimates of basic aerodynamics and control derivatives of the nominal airplane. At this step a moderate-fidelity UAV model was identified.

At the next step of the identification process, wrapping of the traditional technique with the less known Parameter Space Investigation (PSI) method[20] was used to assist in identifying the structure of the regression model for the typical flight regimes. Since the PSI method has been developed to address a correct statement and solution of the multi criteria optimization (identification) problem it provided an ideal tool for the system identification task. The principal advantage of this method consists in the fact that the formulation and solution of the task comprise a single process. At every step of the process, an intellectual input of the designer can be seamlessly integrated to the process therefore modifying its formulation (statement, parameter space, model structure) but preserving and integrating previously acquired data into the final result. The PSI method implementation used for this project is that of the software package MOVI 1.3.[20]

To link MOVI to the Simulink model of the aircraft we employed a previously developed shared-memory interface.[29] This implementation easily allows one to incorporate nonlinear multi-criteria vector optimization/identification into any design task that is implemented in any of the MathWorks' products such as Matlab and Simulink.

C. HIL Architecture

Having solved the autopilot-Simulink communication and developed an accurate *plant model*, the next natural step was to integrate them into the normal operation with the autopilot. In order to achieve this we employed

two different PCs (see Figure 3) named after their main tasks: one in charge of running the Simulink *plant model* **simulation** and the Piccolo Operator Interface (OI) software; and another to run the CAN-UDP **converter** discussed in Section II.A and to display the aircraft status in the Open Source flight simulator FlightGear.[30] It is assumed that both of these PCs are connected to the same network and thus UDP communications is readily available. These components and their interactions are described in the following sections.

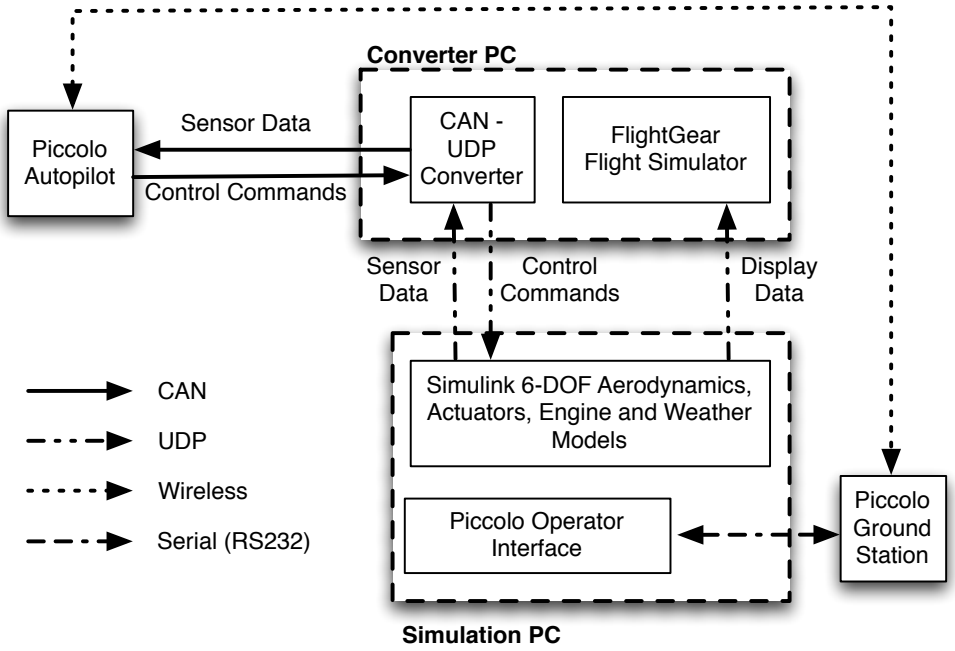


Figure 3: HIL setup; new Simulink-based 6DOF model is a centerpiece of the Simulator

1. The Converter PC

The converter PC has a Pentium 4 microprocessor with 2 GB of RAM running Windows XP operating system. It acts as a proxy between the autopilot and the Simulink *plant model* state. It runs the CAN-UDP converter previously discussed, its front-end GUI is presented in Figure 4. This PC also runs the open source flight simulator FlightGear[30] for display purposes. By harnessing FlightGear's capability to receive plant model data via UDP, the HIL simulator has a high quality graphical display of the UAV status with no programming effort by the end user.

2. The Simulation PC

The simulation PC has a Pentium 4 microprocessor with 2 GB of RAM running Windows XP operating system. It runs the 6DOF aircraft, engine, actuators and atmospheric environment Simulink models at a constant sample period of 0.005 seconds. To facilitate the HIL simulator integration into our current efforts, a Simulink blockset was developed consisting of two components: **(i)** a source which listens on a predefined UDP port and parses messages in order to translate them into the control surface commands (up to 10 different control surfaces); and **(ii)** a sink, with inputs for the traditional sensors (accelerometers, gyros, magnetometers, GPS, dynamic and static pressure, temperature, engine RPM). This sink parses the data and writes it back to a predefined UDP port in the correct format for the CAN-UDP converter.

One of the key capabilities of the autopilot, to echo back a particular received CAN data frame (the *Timing CAN frame*), was used in the HIL simulation. This frame allowed the HIL simulator to send a packet to the autopilot and effectively measure the time it took to travel back. This in turn provided an effective way to monitor the delay between sending sensor data and receiving back a control command

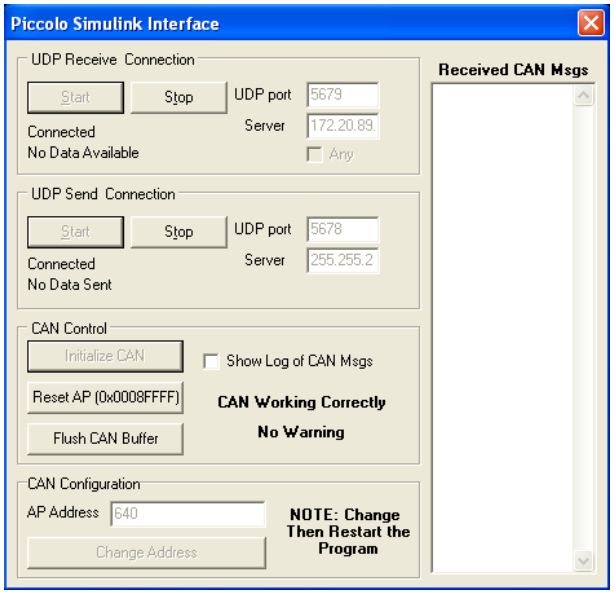


Figure 4: CAN-UDP converter application

associated with that data. During rigorous testing it was observed that this delay was never higher than 0.02 seconds (4 sample periods in the Simulink model).

The same computer also runs the Piccolo OI which is a standard software component of the Piccolo-based setup. The OI provides functionality to configure the autopilot gains, set navigation waypoints, monitor the sensor data and display the UAV in a geo-referenced map. It is connected to the Piccolo Ground Station via serial (RS232 port). More details on the advanced capabilities and versatility of the Piccolo setup can be found in Ref.[10]

It is important to note that this new HIL architecture does not interfere with the standard way of operating Piccolo system. From the Piccolo autopilot's point-of-view, the HIL simulator described in this paper is identical to its original HIL simulator. This allows, for instance, to still run a flight computer onboard as reported in Refs.[7,12,21] offering an added value of higher fidelity flight dynamics modeling to the previously reported flight architectures.

III. Results

This section presents comparative results of two separate experiments using an identified model and the real flight data of Sig Rascal-110 UAV operated by the NPS. The first compared the response of the identified model with that of the real aircraft. The second experiment the compares the frequency response (using Lissajus curves) of the model and the real aircraft. The UAV in flight tests is driven by a sinusoidal reference command of variable frequency by the autopilot; a similar experiment is configured in the new HIL environment and the results are then compared.

A. Doublet Response

The results presented in this Section were obtained as follows: first the Rascal UAV was flown open-loop with doublet commands on the elevator, aileron and rudder channels using the Piccolo autopilot's standard built-in capability. The telemetry data was collected at *100Hz* rate and used to refine the 6-DOF model as described in Section II.B. After completing the system identification phase, the 6-DOF model was run in a software simulation with the exact same doublet commands collected from the flight experiment. Finally, the new HIL simulator was configured and once again ran with the same doublet command data.

Figure 5 shows a comparison between the flight data, the 6-DOF software simulation and the HIL simulation response to the elevator doublet command. The relevant variables for the longitudinal channel

are shown: acceleration in Z-axis (Figure 5.a) , q rate (Figure 5.b) and pitch angle θ (Figure 5.c). Although flight test data is always subject to measurement and environmental noise, it is worth noting how close the HIL dynamics responses (shown in green) resemble those of real flight (shown in blue).

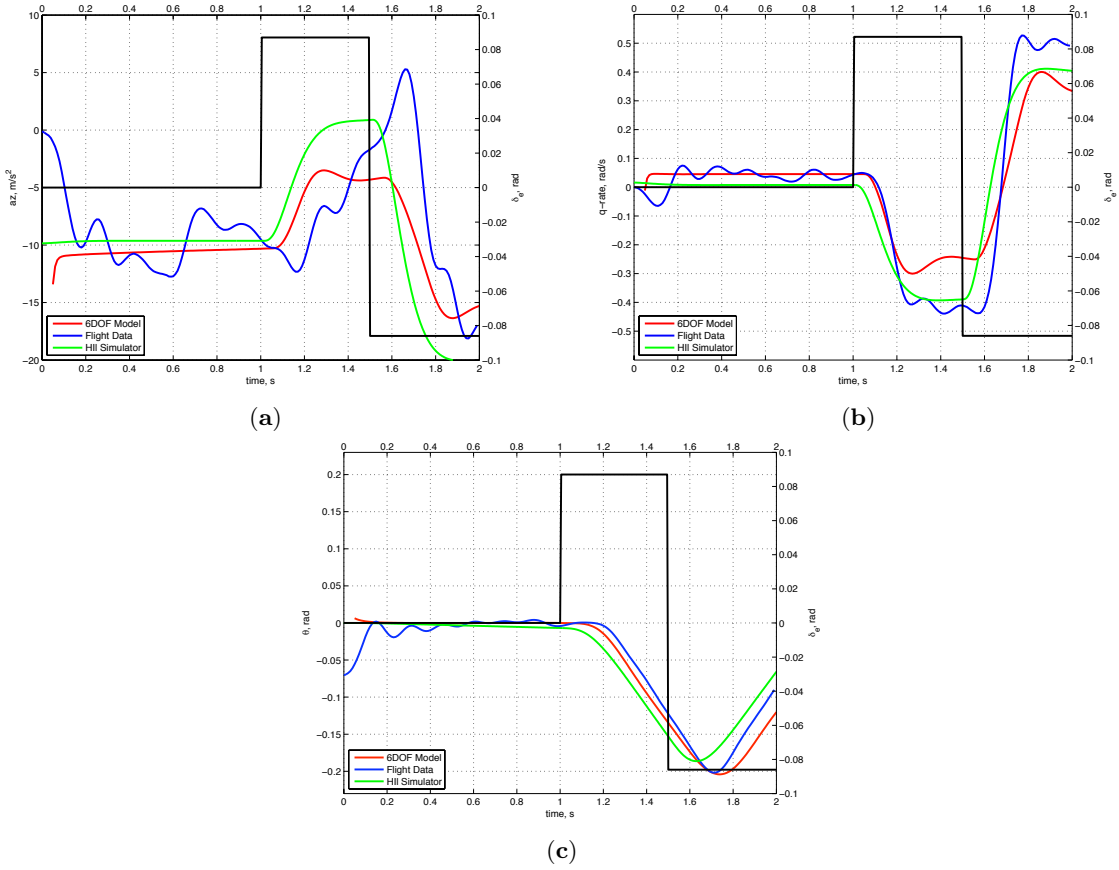


Figure 5: Elevator doublet response. (**a**) Z-axis Acceleration. (**b**) Rotation rate along Y-axis. (**c**) Pitch.

Figure 6 shows the results for the aileron doublet experiment. The relevant variables for the lateral channel are shown: acceleration in Y-axis (Figure 6.a), p rotation rate (Figure 6.b), r rotation rate (Figure 6.c), and yaw angle ψ (Figure 6.d). Worthy of notice is how the HIL results exhibit a *non-minimum phase* response in acceleration in Y-axis, and the p and r rotation rates, similar to that exhibited by the aircraft.

For the lateral channel, system identification was significantly more complicated. The collected data sets showed higher noise than those in the longitudinal channel. Also, as it is well known, the significant coupling between p and r channels make identifying this channel more challenging than the longitudinal one. More flight tests are scheduled to collect better data sets to improve the adequacy of the model.

B. Frequency Response

To explore the frequency response of the aircraft with autopilot when compared with that of the HIL simulator an experiment was set up where the aircraft was controlled from the onboard xPC target PC by a sinusoidal turn rate command consisting of a constant bias turn rate of 6 deg/s (to keep the UAV inside of the closed airspace) and a sine wave with 6 deg/s amplitude and 28 deg/s frequency. The autopilot control loop gains were those of the real airplane for the given flight configuration. The chosen amplitude and bias of the reference sinusoid provide representative and valid comparison avoiding plant saturation; preceding flight tests show the saturation limit of the nominal plant at about 15 deg/s.

The Lissajous *"Input versus Output XY plot"* was used during the course of the experimental studies. The Figure 7 represents the dynamics of a nominal plant explicitly showing the phase shift between the input reference and the output response to be around 92$\pm$ 5 deg for the flight data and 95$\pm$ 7 deg for the HIL results.

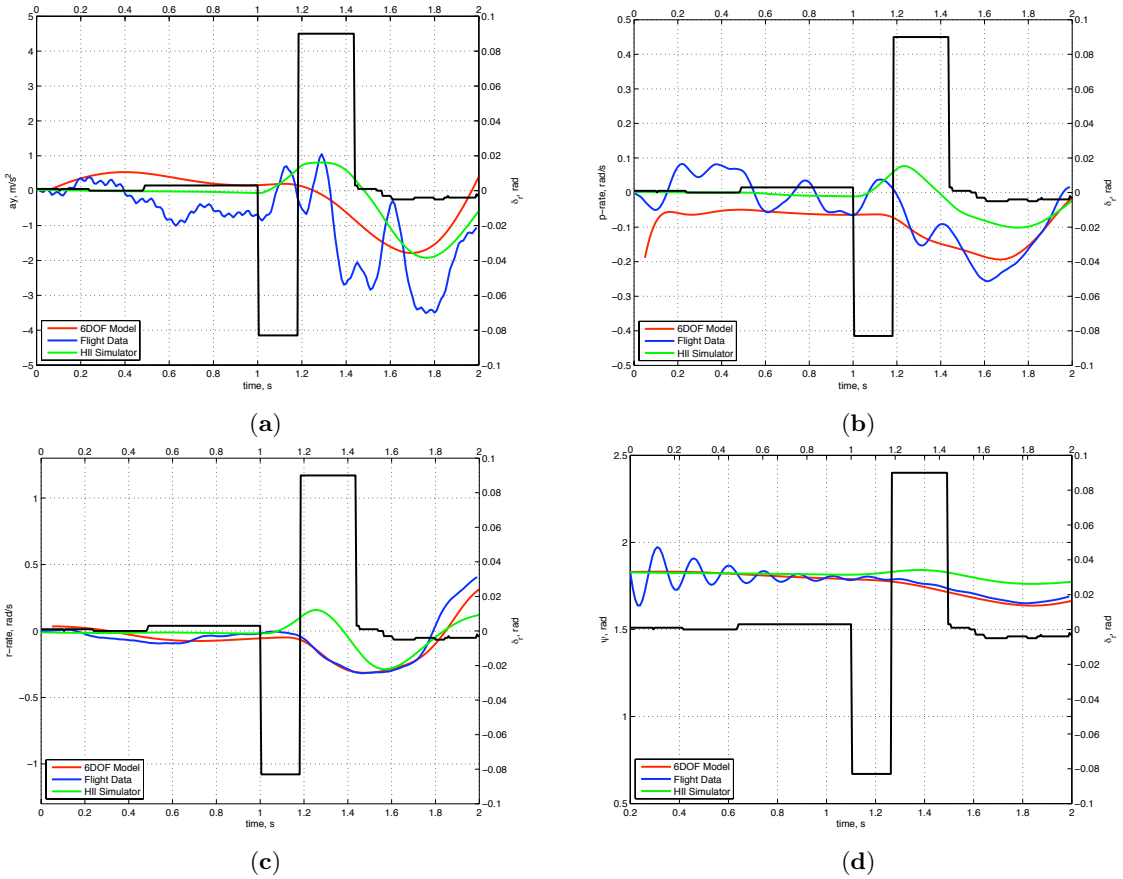


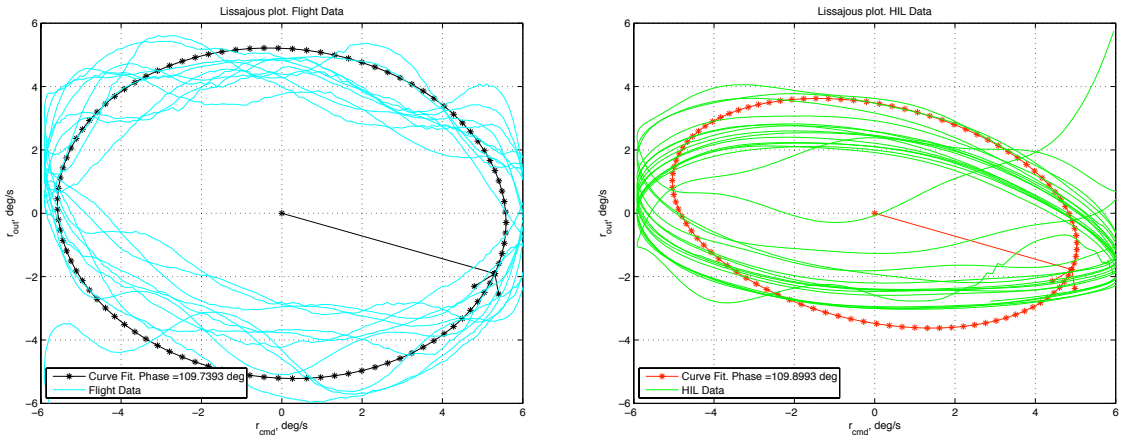
Figure 6: Rudder doublet response. (**a**) Y-axis acceleration. (**b**) Rotation rate along the X-axis. (**c**) Rotation rate along the Z-axis. (**d**) Yaw.

Since the experimental data acquisition is always subject to instrumentation and measurement errors, the black and red ellipses show corresponding averaging fits (in the least square sense) of the frequency response data used to calculate the frequency response. An extended sequence of similar experiments were performed varying the frequency of the reference sinusoid that finally allowed the computation of the frequency response of the Rascal UAV, revealing a close match of the primary stability characteristics.

IV. Conclusions and Further Work

This paper presents a new HIL simulator architecture for the Piccolo autopilot. The primary advantage is achieved by making use of the autopilot's built-in capability to receive simulated sensor data and send control commands via a CAN Bus interface. The new HIL simulator uses Simulink as its main simulation engine offering modeling advances, flexibility, and modification capabilities not available in the provided UAV HIL simulator. By using Simulink, this HIL setup leverages all of the MathWork's tools to make the simulation more rigorous and improve fidelity while dramatically reducing the man hours required for development and modification. Results presented demonstrate that, when properly identified dynamics are employed, the new HIL simulator closely resembles the aircraft flight dynamics thus validating it as an important tool during the control algorithm design phase. On the technology side, the results show that investing in relatively simple software modification (converter) allows for harnessing a power of advanced tools and enables a new quality of HIL simulation to be achieved. Although very powerful in its current configuration, some work is currently underway to allow for seamless integration of the newly developed HIL concept to other available autopilots.

Although the presented architecture could be implemented in a single PC, experimental results showed



(**a**) (**b**)

Figure 7: Lissajaus Curves. (**a**) Flight Test . (**b**) HIL Results

that this is not convenient; mainly due to the significant amount of computing resources required by the non-linear 6-DOF model and the FlightGear Simulator. By separating the required tasks into two PCs the HIL simulator is able to perform well without any delays in the data transmission to/from the autopilot.

Aknowledgements

This work has been partially funded by the Mexican National Science and Technology Council (CONACyT).

References

[1]Gomez, M., "Hardware-in-the-loop Simulation," *Embedded Systems Programming*, Vol. 14, No. 13, December 2001.

[2]Kheir, N. and Holmes, W., "On validating simulation models of missile systems," *Simulation*, Vol. 30, No. 4, Jan 1978, pp. 117–128.

[3]Sisle, M. and Mccarthy, E., "Hardware-in-the-loop simulation for an active missile," *Simulation*, Vol. 39, No. 5, Nov 1982, pp. 159.

[4]Werner, S., Buchwieser, A., and Dickmanns, E. D., "Real-time simulation of visual machine perception for helicopter flight assistance," Vol. 2463, SPIE, 1995, pp. 93–101.

[5]Norlin, K., "Flight Simulation Software at NASA Dryden Flight Research Center," *National Aeronautics and Space Administration*, , No. Nasa Technical Memorandum 104351, Jan 1995.

[6]Maclay, D., "Simulation gets into the loop," *IEE Review*, Vol. 43, No. 3, May 1997, pp. 109–112.

[7]Dobrokhodov, V., Kitsios, I.,Kaminer, I., Jones, K., Xargay, E., Hovakimyan, N., Cao, C., Lizarraga, and M.,Gregory, I., "Flight Validation of Metrics Driven L1 Adaptive Control," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[8]Johnson, E. N. and Schrage, D. P., "The Georgia Tech Unmanned Aerial Research Vehicle: GTMax," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, August 2003.

[9]Johnson, E. N., Schrage, D. P., Prasad, J. V. R., and Vachtsevanos, G. J., "UAV Flight Test Programs at Georgia Tech," *Proceedings of the AIAA Unmanned Unlimited Technical Conference, Workshop and Exhibit*, September 2004.

[10]Vaglienti, B., Hoag, R., and Niculescu, M., *Piccolo System Documentation. Version 1.3.2*, Cloud Cap Technologies, April 2005, `http://www.cloudcaptech.com`.

[11]Tisdale, J., Ryan, A., Zennaro, M., Xiao, X., and Caveney, D., "The software architecture of the Berkeley UAV platform," *Proceedings of the Conference on Control Applications*, Jan 2006.

[12]Dobrokhodov, V., Yakimenko, O., Jones, K., Kaminer, I., Bourakov, E., Kitsios, I., and Lizarraga, M., "New Generation of Rapid Flight Test Prototyping System for Small Unmanned Air Vehicles," *AIAA Modeling and Simulation Technologies Conference Proceedings*, 2007.

[13]MicroPilot, "MicroPilot: World Leader in Miniature UAV Autopilots," http://www.micropilot.com, 2009.

[14]Procerus Technologies, "Kestrel Autopilot," http://www.procerusuav.com/productsKestrelAutopilot.php, 2009.

[15]Brisset, P., Drouin, A., Gorraz, M., Huard, P., and Tyler, J., "The Paparazzi Solution," *2nd US-European Competition and Workshop on Micro Air Vehicles*, November 2006.

[16]The Mathworks, Natick, MA, *Simulink User's Guide*, 2008.

[17]Lizarraga, M. I., *Autonomous Landing System for a UAV*, Master's thesis, Naval Postgraduate School, Monterey, CA, USA., March 2004.

[18]Klein, V. and Morelli, E., *Aircraft System identification: Theory and Practise*, 2006.

[19]Jung, D. and Tsiotras, P., "Modelling and hardware-in-the-loop simulation for a small unmanned aerial vehicle," *AIAA Infotech at Aerospace*, 2007.

[20]Statnikov, R., *Multicriteria Design: Optimization and Identification*, Kluwer Academic Publishers, 1999.

[21]Dobrokhodov, V. and Lizarraga, M., "Developing Serial Communication Interfaces for Rapid Prototyping of Navigation and Control Tasks," *AIAA*, Vol. 6099, 2005, pp. 2005.

[22]The Mathworks, Natick, MA, *xPC Target User's Guide*, 2008.

[23]Vaglienti, B., *Communications for the Piccolo avionics. Version 1.3.2*, Cloud Cap Technologies, September 2006, `http://www.cloudcaptech.com`.

[24]Lawrenz, W., *CAN system engineering: from theory to practical applications*, Springer, 1997.

[25]SYS TEC electronic GmbH, *USB-CANmodul GW-001 Systems Manual*, Germany, October 2008.

[26]Postel, J., "User Datagram Protocol. RFC 768," *Internet Requests for Comments*, 1980.

[27]Desktop Aeronautics, P.O. Box 20384, Stanford CA 94309, *LinAir User's Guide*, 2005.

[28]Morelli, E. and Klein, V., "Application of System Identification to Aircraft at NASA Langley Research Center," *Journal of Aircraft*, Vol. 42, No. 1, 2005, pp. 12–25.

[29]Dobrokhodov, V. and Statnikov, R., "Multi-Criteria Identification of a Controllable Descending System," *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*, 2007, pp. 212–219.

[30]Perry, A. R. and Olson, C., "The FlightGear flight simulator: history, status and future," LinuxTag Conference and Expo, July 2001.