Model-Based Design of UAV Autopilot Software

Zuo Ming[1],Liu Ying[1],Qian Yi[1], Hu Xiongwen[1],
Zhao Xiaochuan[1]
Beijing Institute of Computer Application
Beijing, China
zuoming020@gmail.com

Wang Jinhua[2]
China academy of ordnance science
Beijing, China

***Abstract*—This paper presents a Model-Based approach to develop UAV (Unmanned Aerial Vehicle) autopilot software. It employs Simulink to design the flight controller, Stateflow to implement control logic and Matlab coder to automatically generate embedded C code from the model developed. Software in the loop (SIL) and hardware in the loop (HIL) simulations are performed in the laboratory to validate the software developed. Flight trial cost and risks are minimized and the design cycles are greatly shortened. The feasibility and the effectiveness of the approach are verified through results from lab simulations and field trials.**

Keywords-UAV; model-based design; flight control; simulation)

I. INTRODUCTION *(HEADING 1)*

Model-based design is a modern design methodology that enables faster, more cost-effective development of dynamic systems. Usage of Model-based design in UAV autopilot design has the potentiality to shorten the design cycles and reduce the development cost.

Reference [1] proposed an integral model based design environment for flight control system which emphasized the utilization of Matlab tools during the earlier stage of the development. Reference [2-3] modeled the SUAV's control logic using Stateflow. Reference [4] generated codes from Simulink models using the RTW toolkit to conduct numeric simulations. In [5], HIL simulations were utilized to realize rapid development of a UAV autopilot system.

This paper presents a Model-Based approach to develop UAV autopilot software. Modeling and simulation techniques are combined to accelerate the development.

II. MODEL-BASED DEVELOPMENT AND SIMULATION OF UAV AUTOPILOT SOFTWARE DESIGN

The model-based design approach to develop UAV autopilot software is depicted in Fig 1.

A. Build the Algorithm Model

Simulink and Stateflow are utilized to develop attitude, altitude and throttle controllers and implement flight control logic and navigation algorithm

Simulink is an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems. The attitude controller, altitude controller and throttle controller of the UAV is developed using Simulink.

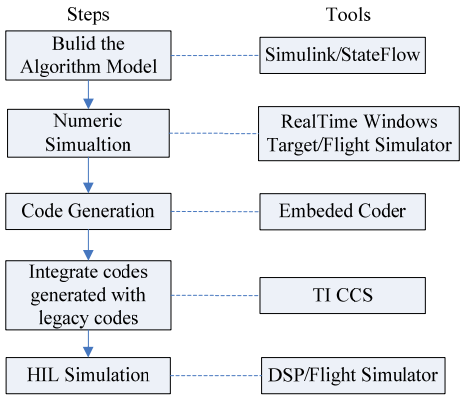


Figure 1 The model-base approach to develop UAV Autopilot Software

During a typical UAV mission, there are multiple flight modes. It requires different control and navigation algorithms be taken during different flight modes and complex logics be embodied for mode transmissions. It is difficult to model these behaviors in Simulink.

Stateflow is a graphic tool based on the finite state machine theory which model complex control logic in a natural, readable and understandable way. It is closely integrated with Matlab\Simulink and suitable to implement the flight mode control logic and navigation algorithm.

A comprehensive, production grade UAV flight control system is developed in Simulink/Stateflow, as presented in Fig 3. All control logics during typical UAV flight missions are taken into account, including automatic takeoff and landing. The ThrottleControl, AltitudeControl and AttitudeControl blocks are simulink subsystems which model the throttle controller, the altitude controller and attitude controller respectively. As this paper emphasizes on the model base design and simulation methodology, the details of the controllers will not be covered.

The FligthManagement block is a Stateflow block which implements the flight control logics. The flight modes include manual flight, automatic takeoff, takeoff waypoints, waypoints, fly to waypoint, hover, return to launch, automatic land, and etc, as shown in Fig 2. The rounded blocks represent different flight modes of the UAV. The transitions between different flight modes according to user commands and flight status are represented as arrowed lines in the chart. The codes on the transition lines indicate the condition or event that triggers the transition and the action that should be taken when the transition occurs.

Published by Atlantis Press, Paris, France.

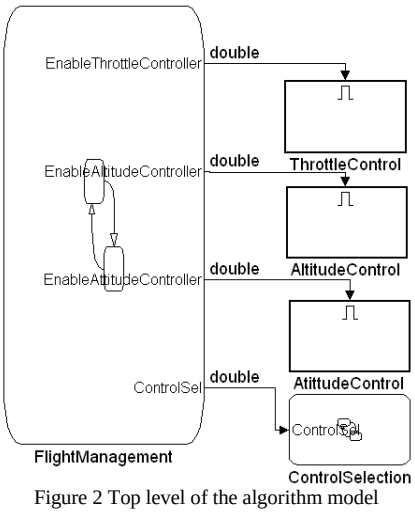


Figure 2 Top level of the algorithm model

B. *Numeric Simulation*

Numeric simulations provide a way to analyze and assess the control effect of the autopilot algorithm during earlier stage of the development period. A UAV autopilot simulation system was designed with a numeric simulation configuration and a hardware-in-the-loop simulation configuration and it was proved to be accurate and easy to use. Commercial off-the-shelf flight simulator software were adopted as flight dynamic modeler as they came with accurate flight models and detailed three dimensional visual effects. Developers were relieved from developing the same functionality in-house which would demand great effort.

The Real-Time Windows Target toolbox of Matlab provides a real-time engine for executing Simulink models on Windows PCs and blocks that connect to a range of I/O boards that are useful for rapid prototyping of UAV autopilot software. Communication between flight simulators and Simulink was established with the Packet in and Packet out blocks to conduct real-time numeric simulations of the autopilot algorithm. The architecture of the autopilot simulation system with numeric simulation configuration is shown in Fig.4.

C. *Code Generation*

The Matlab code generation tools are capable of generating embedded C code optimized for specific hardware directly from Matlab codes and Simulink models. Code generation effectively avoids the possible introduction of manually coded errors and increases the reliability of the code and the consistency between the code implementation and Simulink model.

As the preparation for code generation, the model and its configuration should be adjusted to satisfy the requirement of the code generation tool and the design specification of the autopilot software, such as discretizing the model according to the sample time, setting the discrete solver, configuring the hardware implementation and setting the code style customization options.

In this paper, sample time is set to 0.02.Hardware implementation is set to Texas Instrument C2000 as what is used in the autopilot hardware. The code generation "System target file" is set to "ert.tlc" that represents "embedded coder target".

To start the code generation process, right click the autopilot subsystem and select "Build subsystem…" in the context menu.

D. *Integrate codes generated with legacy codes*

It is not practical or economical to build a model that includes everything of the autopilot software. For example, some functionality such as communication protocol interpreter and specific hardware drivers are easier to write in C code than model in Simulink. The developer may have some legacy codes that are ready to use for which no modeling work need to be done. In the work related to this paper, DSP processors from Texas Instruments are used as the onboard computer. TI provides demonstration programs to accelerate the users' develop process. Under similar circumstance, the handwritten C codes are integrated directly with the code generated.

E. *HIL simulations*

The integrated code is then planted into the autopilot hardware and HIL simulations are conducted to validate the code generated. The architecture of the autopilot simulation system with hardware-in-the-loop configuration is shown in Fig.5. The interface program interprets and forwards the communication packets between the flight simulator and the autopilot hardware to accomplish the transmission of the flight state data and flight control actuator commands. Fig.6 shows the hardware configuration used in the HIL simulation: the three computers, from left to right, are the ground control station, the program computer and the flight simulator respectively. The autopilot hardware is in the bottom left corner.

III. SIMULATION AND EXPERIMENTAL RESULTS

Results from numeric simulations, HIL simulations and flight trials are compared to verify the effectiveness of the model-based design methodology presented.

The wind speed is set to 5m/s during numeric simulations and HIL simulations. The wind intensity measured during flight trials is 3, which indicates a wind speed of 3.4 – 5.4m/s. The airframe used in simulations is Great Plane PT-60 of which the wing span is 1.8 meters and the target airspeed is set to 70 km/h. The airframe used in flight trials is a 1:3 scale model of Piper J-3 of which the wing span is 3 meters and the target airspeed is set to 80km/h.

A screenshot of the flight simulator software taken during simulation is shown in Fig 7. The flight trajectories of numeric simulations, HIL simulations and flight trials are shown in Fig 8-Fig 10. During numeric simulations, the trajectories are plotted real time in Matlab. The HIL simulation trajectories are screenshots taken from the ground control station. The flight trial trajectories are plotted in Matlab with data recorded during flights by an onboard data logger.

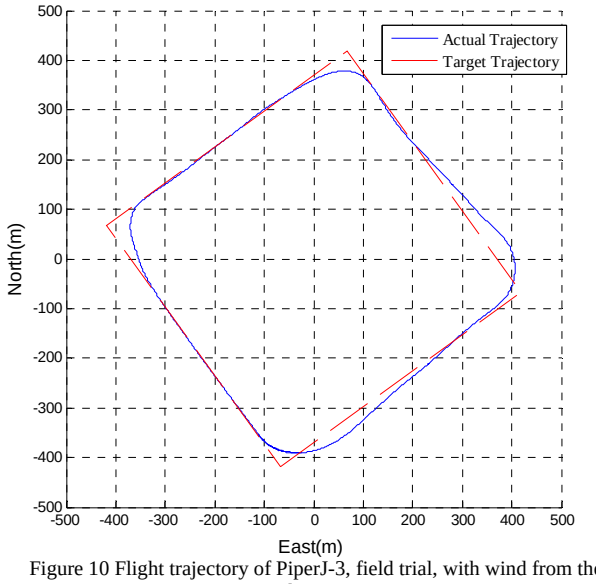


Figure 10 Flight trajectory of PiperJ-3, field trial, with wind from the southwest

As shown in Fig.8–Fig.10, when the airplanes turn from the upwind side to the downwind side, the trajectories during the turn fall inside the corner of the desired path and when the airplanes turn from the downwind side to the upwind side, the trajectories fall outside the corner. The flight control software developed shows great consistency during numeric simulations, HIL simulations and flight trials.

IV. CONCLUSION

The development process of a comprehensive, production grade UAV flight control system is presented. Embedded codes are automatically generated from Simulink models. Numeric and HIL simulations are conducted to verify the autopilot software. Simulation and field trial results show that the simulations accurately predict the real-life behavior of the autopilot software and the code generation process is effective. The model-based design approach is proved to be feasible and effective.

REFERENCES

[1] Yang Xiangzhong, Cui Wenge. Research on Flight Control System Integral Design Based on Model[J], Journal of System Simulation, 2007, 19(19):4411-4416

[2] Zhang Zhisheng, Chen HuaiMin. The Modeling and Simulation of SUAV's Control Logic[J], Fire Control & Command Control, 2010, 35(9):93-97

[3] Li Junmei, Cheng Yongmei,. Research of multi-mode flight simulation based on Stateflow [J], Application Research of Computers, 2011, 28(12):4557-4559

[4] Wang Geng, Jia Wei. Implementation of One UAV Flight Control Testing System Based on Simulink and DSP[J], Journal of Projectiles, Rockets. Missiles and Guidance, 2008, 28(1):286-288

[5] Widyawardana Adiprawita, Adang Suwandi Ahmad.Hardware In The Loop Simulator in UAV Rapid Development Life Cycle, International Conference on Intelligent Unmanned Systems (ICIUS), 2007 Indonesia, 31-36

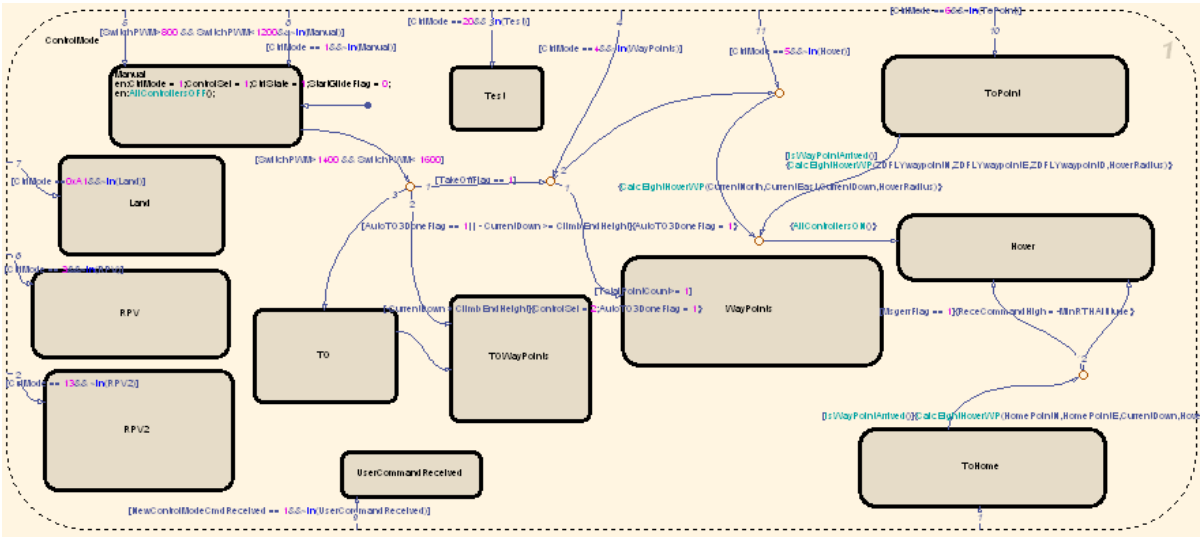


Figure 3Stateflow chart for the UAV control logic

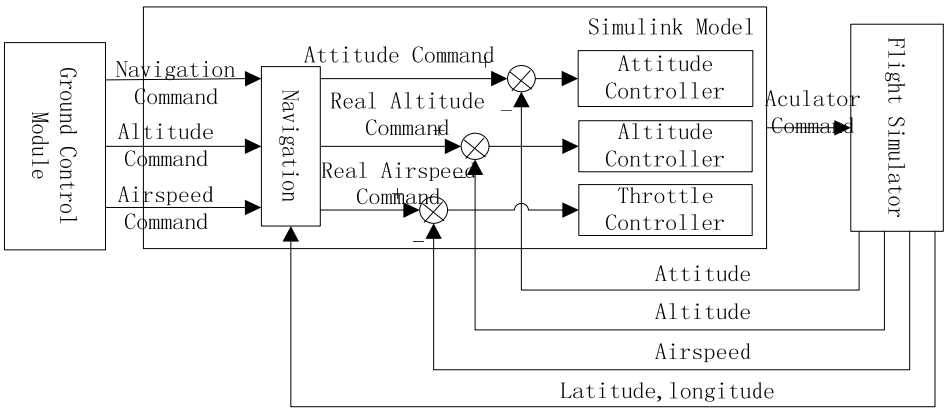


Figure 4 Numeric simulation system architecture

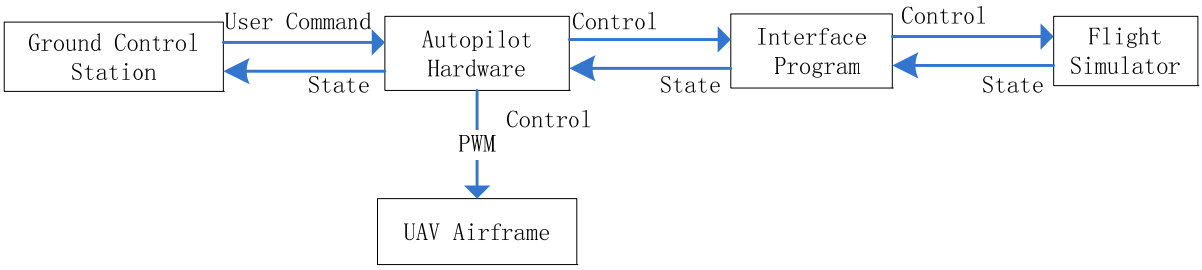


Figure 5 HIL simulation system architecture

Figure 6 Picture of the HIL simulation environment

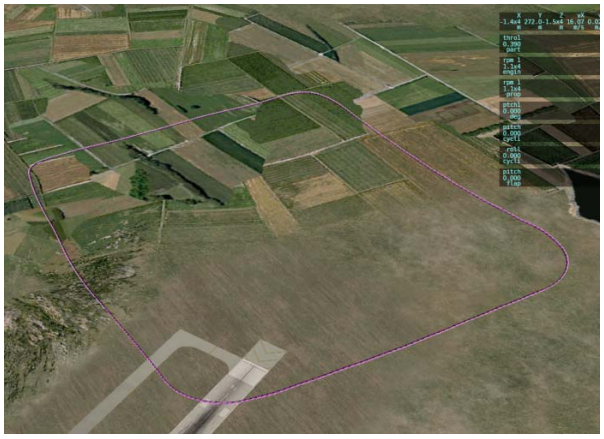


Figure 7 Screenshot of the flight simulator during a simulation

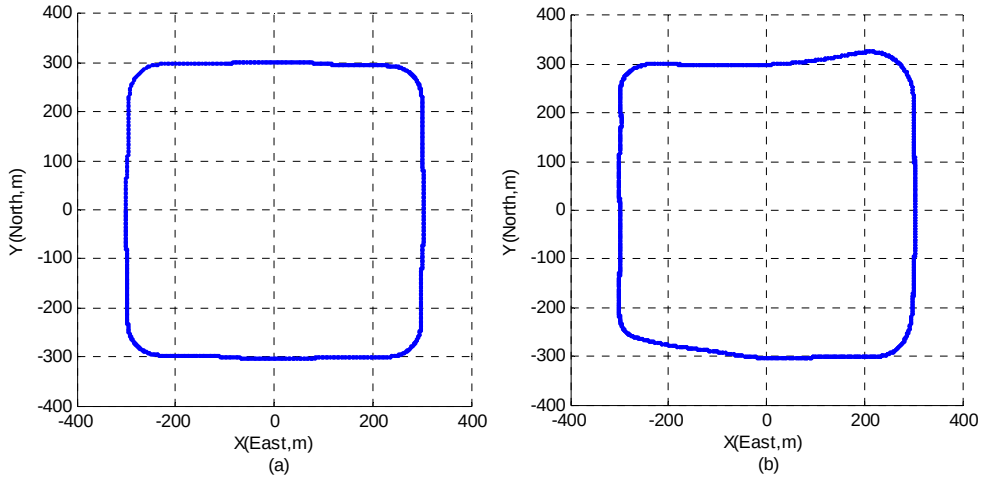
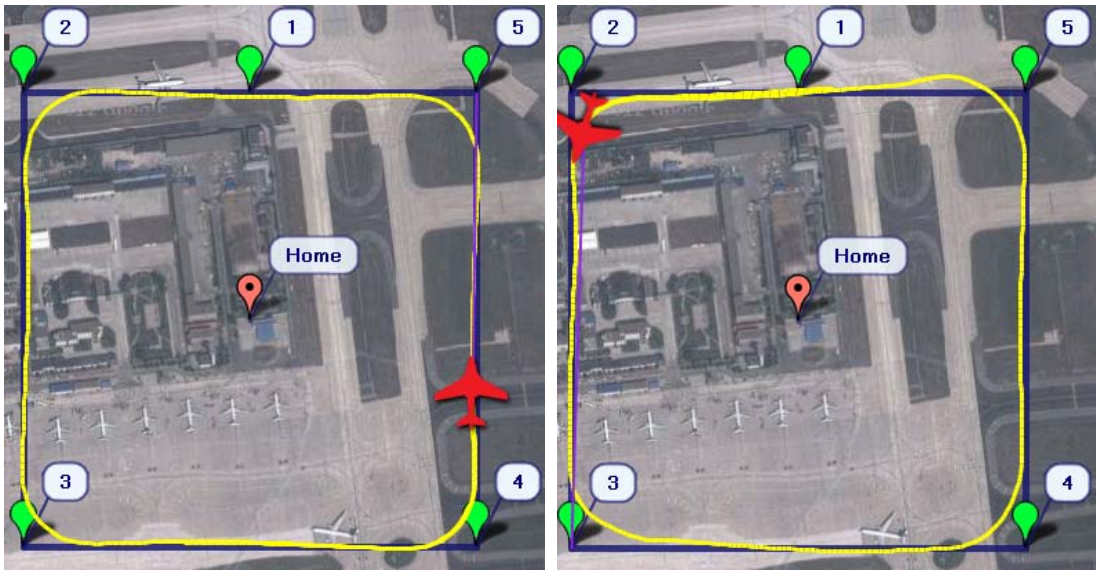


Figure 8 Flight trajectory of PT-60, numeric simulations:(a)without wind;(b)with wind, 5m/s from the southeast.



(a) (b)

Figure 9 Flight trajectory of PT-60, HIL simulations : (a)without wind;(b)with wind, 5m/s from the southeast.