Hardware-In-the-Loop Simulation of UAV for Fault Injection

Siyang Gong[1], Shengwei Meng[1,2], Benkuan Wang[1,2], Datong Liu[1,2,*]
1 Automatic Test and Control Institute, Harbin Institute of Technology, Harbin, China
2 School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China
E-mail: liudatong@hit.edu.cn

***Abstract*—Unmanned aerial vehicles (UAVs) are widely used in military and civilian applications. The safety of UAV has been paid more and more attention. Prognostic and Health Management (PHM) of UAV can realize the fault prediction during flight and make an appropriate response according to potential faults. Therefore, the safety and reliability of UAV are improved. However, the existing PHM research on UAVs has the following two challenges: a) the number of fault samples in historical flight data is small, and it is impossible to cover multiple fault modes of UAVs to meet the demand of modeling and verification, b) the actual flight verification of the UAV PHM technology cannot be carried out directly through actual flight test. Aiming at the above problems, this paper proposes a fault data generation method based on the Hardware-In-the-Loop Simulation (HILS) technology. We construct a UAV model and its corresponding fault models by analyzing fault features of UAV flight control system. It can simulate the actual faults and generate flight data with multiple fault modes, thereby available data are provided to support PHM research. Besides, PHM algorithms implemented on airborne PHM modules can be verified via this platform in real-time. The experiment results show that the HILS platform has a good performance for fault injection.**

Keywords—UAV; HILS; Fault Injection

I. INTRODUCTION

With the development of unmanned aerial vehicle (UAV) technology, the airborne equipment of modern UAV systems is becoming more and more complicated. Improving the safety, reliability, efficiency of use and reducing the cost of use and maintenance has become a hotspot in current research [1][2]. Prognostic and health management (PHM) could monitor the flight state of UAV and predict the potential fault, which is useful in condition-based maintenance [3][4]. Through the PHM technology [5][6], UAV can predict future performance [7]-[9] and potential failures [10]. Therefore, the necessary maintenance resources will be prepared, thus reducing the cost during the life cycle.

However, the existing research of PHM technology and methods on UAV faces the following two challenges: 1) the number of fault samples in historical flight data is small, and it is impossible to cover multiple fault modes of UAVs to meet the demand of modeling and verification on PHM research. 2) It is difficult to carry out flight verification of the PHM technology on the UAV through actual flight test [11]. In summary, it is essential to generate fault data of sensor and actuator for UAV PHM research.

Fault injection technology is an effective method for PHM validation [12][13], which refers to artificially generating faults to be applied to a specific target system according to the specified fault type, thereby accelerating the failure and error of the system. In the meantime, the system response to the injected faults is observed and collected, about which the information can be analyzed to provide PHM researchers with fault data. Fault injection can be implemented by HILS [14]. In addition to generating fault data, HILS can achieve communication between the simulation model and the flight controller. This means that we can verify our PHM algorithms running on airborne PHM hardware module in a real-time way.

In this paper, a hardware-in-the-loop simulation platform is constructed for sensor and actuator fault injection of UAV flight control system. UAV airframe, sensors and actuators are the simulation parts, and Pixhawk is the real hardware part. On the platform, research on fault injection and fault data generation will be carried out by analyzing fault modes of sensors and actuators. Our fault injection method can simulate the fault occurrence time, the fault location and fault mode randomly, thus obtaining fault data in different fault modes and different time periods and different parts. Our research solves the issue of the lack of fault samples on UAV and provides data support for UAV PHM research.

The remainder of this paper is organized as below. Section II analyzes the fault features of sensors and actuators, and their mathematical models are established. Section III shows the architecture and implementation of HILS platform. Section IV presents the result of HILS and fault injection. Section V concludes the paper.

II. FAULT FEATURES ANALYSIS

A. Analysis of Sensor Fault Features

The sensor is one of the parts that often fails during the flight of the UAV. We analyze the features of the sensor before and after its failure which will facilitate the classification and modeling of sensor faults. Therefore, the foundation for subsequent fault simulation and fault diagnosis will be laid. In this paper, according to the cause of sensor fault, it is divided

This work is supported partly by Natural Science Foundation of China under Grant No. 61803121 and 61771157.

978-1-7281-0860-5/19/$31.00 ©2019 IEEE

into constant deviation fault, gradual drift fault, precision damage fault, stuck fault and gain change fault. We will analyze their features and model the above sensor faults.

There is a certain difference between the output of the sensor after a constant deviation fault and the normal output, which is caused by the internal bias voltage or bias current. Gradual drift fault means that a time-varying difference exists between the fault output and the normal output and it is used to simulate a fault caused by the temperature drift of the device. Precision damage fault means that the output under fault fluctuates up and down on the basis of the normal output and it is used to simulate a fault caused by the decrease in the measurement accuracy of the sensor. The output of the stuck fault is a constant that does not change with time and it is used to simulate a fault caused by mechanical jamming of the internal components of the sensor. The output of the sensor with short-circuit fault is always zero, which is used to simulate the fault caused by bridge corrosion and short circuit connection, etc. Figure 1 shows the manifestation of various types of sensor faults.

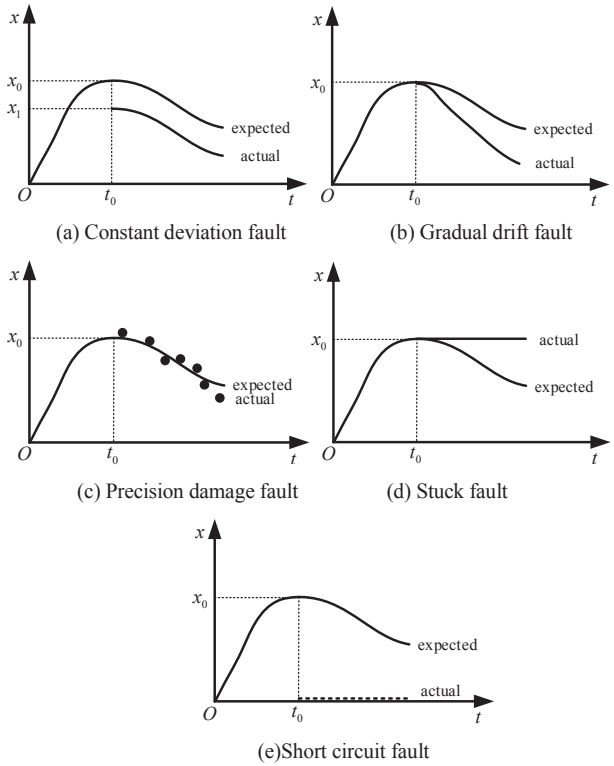

Figure 1. Feature diagram of sensor fault

Through the above analysis of sensor fault modes, the mathematical model of sensor fault is represented as follow:

$$y_m(k) = s(k)\,y_c(k) + d(k) \tag{1}$$

where $y_c(k)$ is the given expected output value and $y_m(k)$ is the actual output value of the sensor. $s(k)$ represents gain between the sensor's output and input. k represents the current moment. $d(k)$ is the deviation from the output value. Table I shows the sensor fault mode and its corresponding parameter values [15].

TABLE I. SENSOR FAULT MODE

Fault Mode	Value of $s(k)$ and $d(k)$
Fault free	$s(k)=1$, $d(k)=0$
Constant deviation fault	$s(k)=1$, $d(k)$ is a fixed value
Gradual drift fault	$s(k)=1$, $d(k)$ is a value varying with time
Precision damage fault	$s(k)\neq 1$, $d(k)$ is a value varying with time
Stuck fault	$s(k)=0$, $d(k)$ is a fixed value
Short circuit fault	$s(k)=0$, $d(k)=0$

B. *Analysis of Actuator Fault Characteristics*

Similar to the sensor fault classification, actuator fault is divided into stuck fault, damage fault, loose fault and constant deviation fault. In this paper, research on fault manifestation is carried out. We analyze the characteristics of stuck fault, damage fault, loose fault and constant deviation fault of actuators. And then we establish the corresponding mathematical model.

Stuck faults refer to that the output of steering gear stuck at a certain position and do not change anymore. After this fault steering gear no longer responds to the control instructions of the flight control computer. This kind of fault is caused by mechanical stuck. Damage faults are used to simulate the reduction of output gain of steering gear and worse control effect. Loose fault means that actuators do not deflect according to flight control computer's command any more, but to deflect freely without rules, and cannot generate corresponding torque. Constant deviation fault means that there is a constant deviation between the actuator output and the desired output and the deviation does not change with time. Figure 2 shows the manifestation of various types of actuator faults.

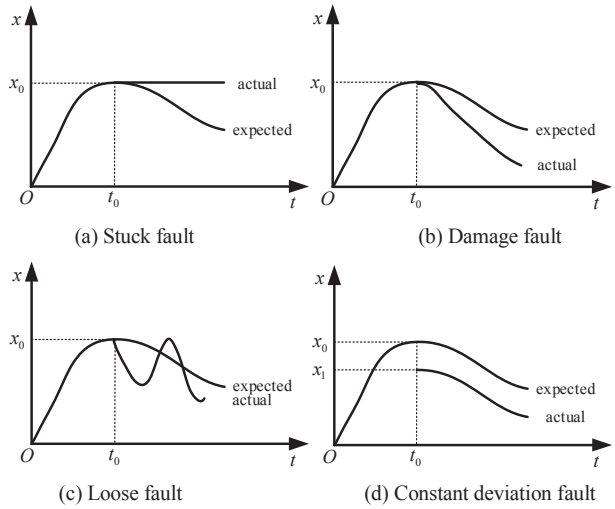

Figure 2. Feature diagram of actuator fault

The mathematical model of actuator fault is represented as follow:

$$u_m(k) = s(k)u_c(k) + d(k) \tag{2}$$

where $u_c(k)$ is the given expected output value and $u_m(k)$ is the actual output value of the actuator. $s(k)$ represents the gain between actuator's output and input. k represents the current moment. $d(k)$ is the deviation from the output value. Table II shows the sensor fault mode and its corresponding parameter values [15].

TABLE II. ACTUATOR FAULT MODE

Fault Mode	Value of $s(k)$ and $d(k)$
Fault free	$s(k)$=1 , $d(k)$=0
Stuck fault	$s(k)$=0 , $d(k)$ is a fixed value
Damage fault	$s(k) \subset (0,1)$, $d(k)=0$
Loose fault	$s(k)=0$, $d(k)$ varies freely
Constant deviation fault	$s(k)$=1 , $d(k)$ is a fixed value

III. HILS DESIGN AND IMPLEMENTATION

A. Architecture of HILS Platform

Our hardware-in-the-loop simulation platform for the flight control system of UAV consists of the simulation control computer, the simulation computer and the flight control computer, as shown by Fig. 3.

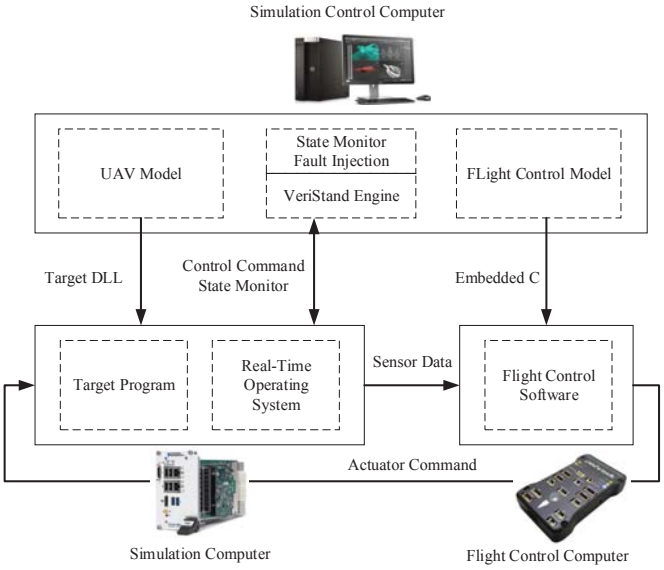

Figure 3. Diagram of HILS platform

1) Simulation Control Computer: Matlab/Simulink and NI VeriStand Software are installed on this computer. We establish a UAV simulation model and the corresponding flight control model in Simulink. And then we convert the UAV model to an embedded target program that can be executed in real-time operating system and transplant the target program to the simulation computer. Similarly, we do the same thing for the flight control model. A human-machine interaction interface is built to monitor the running status of HILS, control the simulation process and implement fault injection by VeriStand Engine.

2) Simulation Computer: The UAV simulation model runs on this computer. It receives control command of actuators that flight control computer sends and calculates the flight state parameters. The flight state parameters are regarded as sensor data and the simulation computer sends sensor data to the flight control computer.

3) Flight Control Computer: The flight control model is executed on this computer. It receives sensor data transmitted by the simulation computer. The flight control software calculates the desired control command of actuators and sends it to the simulation computer.

B. Implementation of HILS platform

Our model for HILS is based on the model of "*UltraStick 25E*" UAV that University of Minnesota Digital Conservancy has constructed [16]. A summary of important physical parameters is provided in Table III.

TABLE III. SUMMARY OF AIRCRAFT GEOMETRY

Parameter	Value and Units
Wing reference area	0.32 m^2
Wing span	1.2 m
Wing chord	0.3 m
Weight	1.9 kg

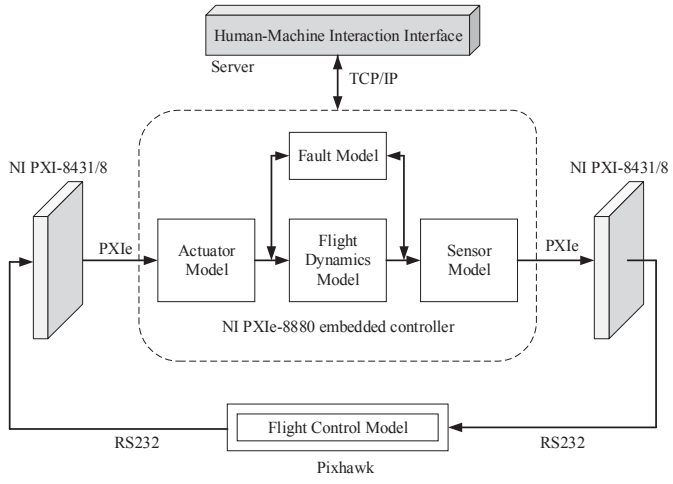

Figure 4. Implementation diagram of HILS

Pixhawk, an open-source professional autopilot hardware is chosen as our flight control computer, on which flight control software is running. The NI PXIe-8880 embedded computer is a high bandwidth PXI Express system controller and has an eight-core Intel Xeon E5-2618L v3 processor, which is used to

execute the simulation model consisting of actuator model, flight dynamics model, fault model and sensor model. A server used as a simulation control computer is responsible for monitoring the system operating state, controlling fault injection and managing fault data generated by the fault model. Figure 4 shows our implementation diagram.

Based on the Simulink model provided by the UAV Lab at the University of Minnesota, we design our own model to fit for our UAV as shown by Fig. 5. The final model we have designed could be compiled by NI Model Interface Toolkit and Embedded Coder, thus generating executable code in NI Pharlap real-time operating system.

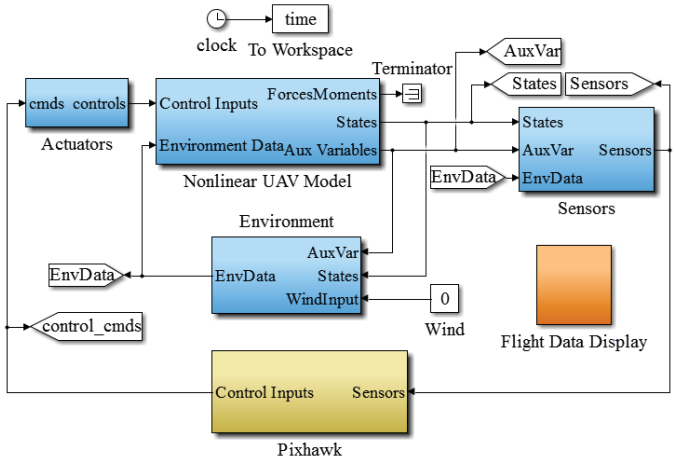

Figure 5. Simulink model of UAV

According to the analysis of fault mode and its characteristics, fault injection is implemented as shown by Fig. 6. Fault injection model obtains data from the actuator model, and then adds fault to the data, finally sends the data containing fault to flight dynamics model. The above process is the actuator fault injection. The sensor fault injection is the same.

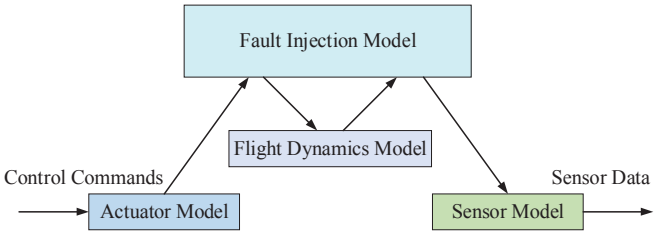

Figure 6. Implementation diagram of fault injection

IV. Experiments and Results

A. Experiment Discription

As shown by Fig. 7, the comparison experiment between Simulink simulation and HILS is carried out first to prove the correctness of code conversion from the Simulink model to executive code. After that, fault injection for sensors and actuators is implemented on the HILS platform. According to the generated data, we verify the effectiveness of fault injection.

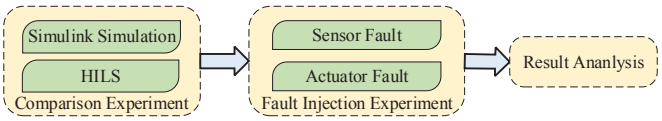

Figure 7. Experiment flow chart

Specifically, in the comparison experiment, we take the lateral movement of UAV as an example, set the simulation time to 50 second and sample time to 0.02 second. We run the UAV model and the flight control model in Simulink and collect the generated data to verify that the model works correctly. Additionally, we convert the model to an executive code and run the model on HILS platform. We compare the flight state data between Simulink simulation and HILS.

Our fault injection experiment is carried out on HILS platform. Limited to the paper volume, we inject stuck fault to the sensor specifically the roll sensor. And as an example of actuators, the left aileron is injected with stuck fault too. Under the above two experimental conditions, we collect the output data of left aileron, right aileron, roll and roll angle rate to analyze the influence of stuck fault.

B. Comparison Experiment Between Simulink and HILS

In this experiment, we run the Simulink simulation first to verify the correctness of our model. Particularly, we give control command to our UAV model and observe the model's output.

Figure 8 shows the control command applied to the model. The blue line is the left aileron deflection command and the red line is the right aileron deflection command. Both of them are the input of the UAV model. Figure 9 shows the change of attitude angle and angular rate which are the output of the UAV model. Comparing Fig. 8 and Fig. 9, we can find that the change of roll angular rate is consistent with that of left aileron. It means that our model has a good performance.

After the verification of the UAV model, we execute the model both in Simulink and HILS platform. During the simulation, we collect roll angle data, pitch angle data and yaw angle data. Figure 9 represents the result of Simulink simulation and Fig. 10 indicates the HILS's result. By comparing the simulation result in the following two figures, the code conversion from Simulink model to executive code is pretty good.

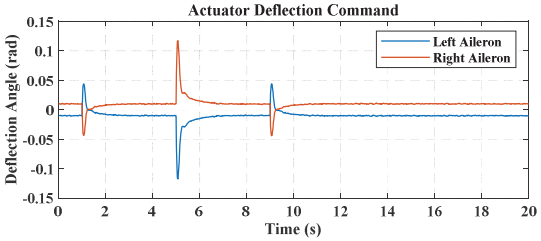

Figure 8. Actuator control command

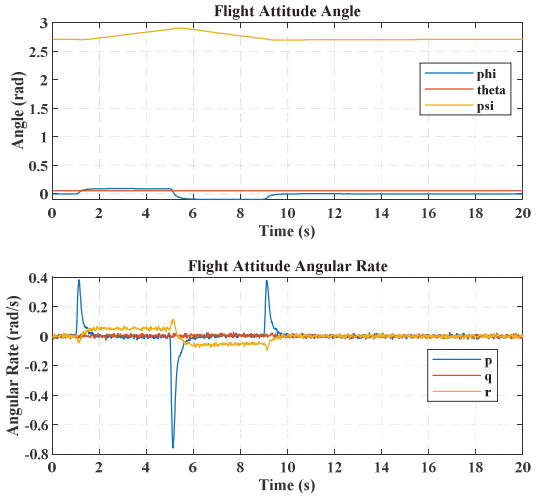

Figure 9. Simulated flight data

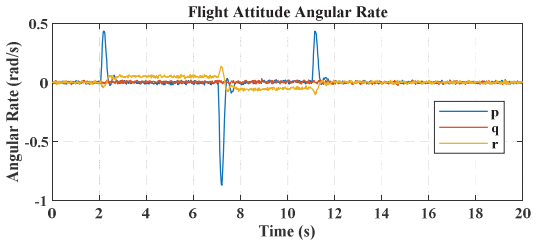

Figure 10. Attitude angle under HILS

C. *Fault Injection Experiment on HILS Platform*

As illustrated in section A, we inject stuck fault to roll sensor and left aileron on HILS platform. Taking lateral movement of UAV as an example, when the simulation runs to 6 seconds, a stuck fault is injected. After the stuck fault, we collect the output data of left aileron, right aileron, roll sensor and roll angular rate sensor to verify that the fault injection works effectively.

Carrying out on fault injection of roll sensor, we collect sensor data and actuator data. Figure 11 shows the output of left aileron and right aileron under normal and fault conditions. The blue line represents the normal output and the red line indicates the output after the fault occurs. The deflection angle of left aileron and right aileron have an increasing deviation from the normal output at a fixed rate and finally approach 0.1383 radians and minus 0.1383 radians apart. The result illustrates the influence of roll sensor stuck fault towards actuators.

Figure 12 shows the change of roll and its rate after the stuck fault. In the upper figure of Fig. 12, the blue line is the roll control command, the red line represents roll output under normal conditions and the orange line indicates roll output under fault conditions. The output of roll follows input well under normal circumstances. After the stuck fault at 6 seconds, the output keeps a five-degree deflection angle and never changes anymore verifying that the system has a stuck fault of roll. The lower figure illustrates the change of roll angular rate after the fault occurs.

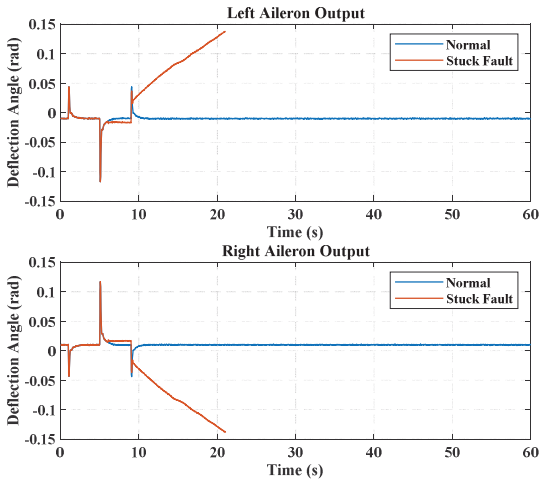

Figure 11. Output of ailerons after roll sensor's stuck fault

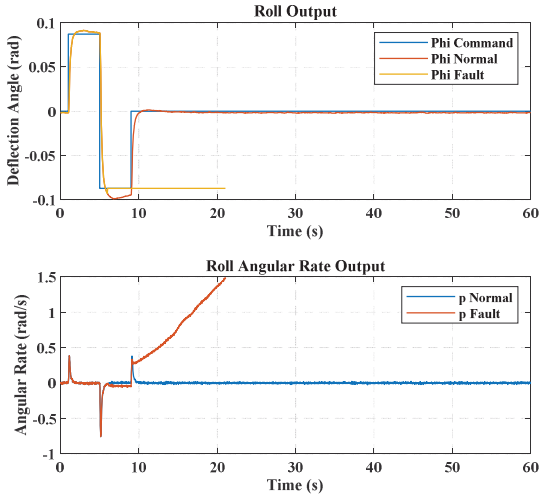

Figure 12. Influence on stuck fault of roll sensor towards flight state

Fault injection of the actuator is the same as that of the sensor. We inject stuck fault to the left aileron during the simulation. And the above two figures show the result.

Figure 13 indicates the output of left aileron and right aileron under normal and fault conditions. The blue line represents the normal output and the red line indicates the output after the fault occurs. The deflection angle of left aileron jumps to minus 0.0151 radians and the right aileron is just the opposite. This denotes that the left aileron has a stuck fault at minus 0.0151 radians.

Figure 14 shows the change of roll and its rate after stuck fault. In the upper figure of Fig. 14, the blue line is the roll control command, the red line represents roll's normal output and the orange line indicates roll output under fault conditions. The output of roll follows input well under normal circumstances. After the stuck fault at 6 seconds, the output of roll has an increasing deviation from the normal output and finally reaches minus -1.157 radians. In the lower figure, the blue line is the normal output and the red line is the output under fault circumstances. Similarly, the output of roll angular rate jumps to nearly minus 0.12 rad/s. Since the ailerons are

used to the roll movement of UAV, once the left aileron has a stuck fault, the output of roll and its angular rate also have a deviation from the normal output.

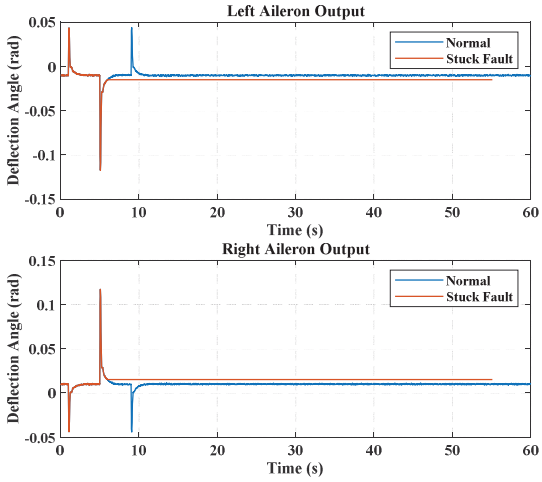

Figure 13. Output of ailerons after stuck fault

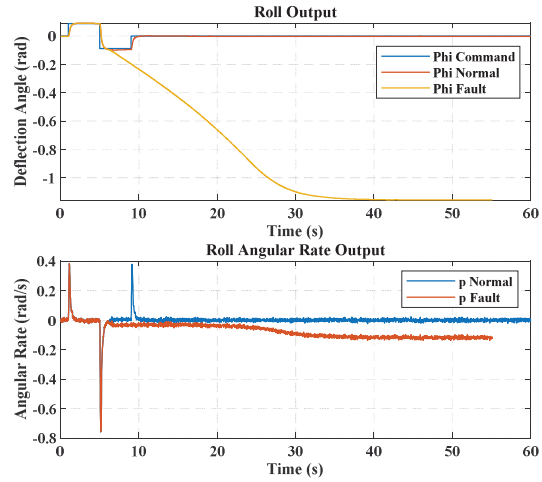

Figure 14. Influence on stuck fault of left aileron towards flight state

V. CONCLUSION

Due to the lack of fault data of UAV, we establish a hardware-in-the-loop simulation platform on which the fault model of UAV can run to generate fault data containing various types of faults. Compared to Simulink simulation, HILS is closer to actual circumstance. More importantly, we can verify our PHM algorithms running on airborne PHM hardware module in a real-time way by HILS technology. First of all, we analyze the fault modes of sensors and actuators, then establish the corresponding mathematical model. Secondly, a HILS platform is constructed which consists of a real Pixhawk autopilot hardware, an embedded controller and a server. Finally, the simulation model is converted to an executive code and runs on the platform to generate fault data. Through the comparison experiment of Simulink simulation and HILS, the effectiveness of our platform used to generate fault data is proved.

Because of the limit of the paper volume, other kinds of fault will also be injected. Our work achieves the goal of generating fault data and provide support for anomaly detection, fault diagnosis and PHM in future work.

REFERENCES

[1] Y. Kang, B. Park, A. Cho, C. Yoo, and S. Koo, "Flight test of flight control performance for airplane mode of Smart UAV," International Conference on Control, Automation and Systems, JeJu Island, pp. 1738-1741, 2012.

[2] N. Snooke and C. Price, "Automated FMEA based diagnostic symptom generation," Advanced Engineering Informatics, vol. 4, Iss. 26, pp. 870-888, 2012.

[3] D. Pan, "Hybrid data-driven anomaly detection method to improve UAV operating reliability," Prognostics and System Health Management Conference, Harbin, pp. 1-4, 2017.

[4] I. P. d. Medeiros, L. R. Rodrigues, C. S. Kern, R. D. C. d. Santos, and E. H. Shiguemori, "Integrated task assignment and maintenance recommendation based on system architecture and PHM information for UAVs," in 2015 Annual IEEE Systems Conference (SysCon) Proceedings, 2015, pp. 182-188.

[5] L. Liu, Q. Guo, D. Liu, and Y. Peng, "Data-Driven Remaining Useful Life Prediction Considering Sensor Anomaly Detection and Data Recovery," IEEE Access, vol. 7, pp. 58336-58345, 2019.

[6] L. Liu, S. Wang, D. Liu, and Y. Peng, "Quantitative selection of sensor data based on improved permutation entropy for system remaining useful life prediction," Microelectronics Reliability, vol. 75, pp. 264-270, 2017.

[7] D. Liu, L. Li, Y. Song, L. Wu, and Y. Peng, "Hybrid state of charge estimation for lithium-ion battery under dynamic operating conditions," International Journal of Electrical Power & Energy Systems, vol. 110, pp. 48-61, 2019.

[8] D. Liu, Y. Song, L. Li, H. Liao, and Y. Peng, "On-line life cycle health assessment for lithium-ion battery in electric vehicles," Journal of Cleaner Production, vol. 199, pp. 1050-1065, 2018.

[9] Y. Song, D. Liu, C. Yang, and Y. Peng, "Data-driven hybrid remaining useful life estimation approach for spacecraft lithium-ion battery," Microelectronics Reliability, vol. 75, pp. 142-153, 2017.

[10] H. Fan, H. Fang, Y. Dong, H. Shi, and S. Ren, "UAV engine fault and diagnosis with parameter models based on telemetry data," in 2017 Prognostics and System Health Management Conference (PHM-Harbin), 2017, pp. 1-6.

[11] X. Li and W. Zhang, "Design of Prognostic and Health Management Structure for UAV System," in 2011 21st International Conference on Systems Engineering, 2011, pp. 12-16.

[12] M. Liu, Z. Zeng, F. Su, and J. Cai, "Research on fault injection technology for embedded software based on JTAG interface," in 2016 11th International Conference on Reliability, Maintainability and Safety (ICRMS), 2016, pp. 1-6.

[13] D. He, N. Hu, and M. Wang, "Study on real-time fault injection and simulation of mechanic-electronic-hydraulic control system based on AMESim and LabVIEW," in 2014 Prognostics and System Health Management Conference (PHM-2014 Hunan), 2014, pp. 446-450.

[14] Y. Liao, X. Shi, C. Fu, and J. Meng, "Hardware in-the-loop simulation system based on NI-PXI for operation and control of microgrid," in 2014 9th IEEE Conference on Industrial Electronics and Applications, 2014, pp. 1366-1370.

[15] K. Guo, L. Liu, S. Shi, D. Liu, and X. Peng, "UAV Sensor Fault Detection Using a Classifier without Negative Samples: A Local Density Regulated Optimization Algorithm," Sensors, vol. 19, p. 771, 2019.

[16] Y. C. Paw, "Synthesis and validation of flight control for UAV," Ph.D. Dissertations, University of Minnesota, America, 2009.