Defence Science Journal, Vol. 64, No. 4, July 2014, pp. 327-331, DOI : 10.14429/dsj.64.4933


# Simulation of Flapping-wing Unmanned Aerial Vehicle using X-plane and Matlab/Simulink

A. Kaviyarasu* and K. Senthil Kumar

*Madras Institute of Technology, Anna University, Chennai-600 044, India*
**E-mail: isrokavi@gmail.com*

**ABSTRACT**

This paper presents the simulation of flapping-wing unmanned aerial vehicle model using X-plane and Matlab/Simulink. The flapping-wing ornithopter model (i.e. an aircraft that flies by flapping its wings) has been developed in plane maker software and executed in the X-plane environment. The key idea of flapping-wing mechanism in X-plane software is by varying its dihedral angle sinusoidally. This sinusoidally varying dihedral angle of wing creates upward and downward stroke moments inturn this creates a lift and a forward thrust for flying the flapping-wing model. Here pitch, roll, yaw and throttle (flapping rate) is fed as reference input through the user datagram protocol (UDP) port. The difference between the reference inputs, the simulated outputs are again fed back to simulator through UDP port and the gains are observed for the responses of flapping-wing unmanned aerial vehicle in Matlab/Simulink environment. Here various gains are used to monitor the optimized flying of flapping-wing model.

**Keywords:** Unmanned aerial vehicle, X-plane, Matlab, Simulink, plane maker, flapping-wing, ethernet

## 1. INTRODUCTION

Flapping-wing flight naturally has an unparalleled maneuverability, agility, and hovering capability. Over the last few decades, engineers have made remarkable progress toward the design of flapping-wing unmanned aerial vehicles known as Ornithopter[1]. Ornithopter is nothing but a flying vehicle that flies like a bird. The heart of the ornithopter is its flapping-wing mechanism, it converts the rotating motion from the brushless motor into a flapping motion.

The Fig. 1 shows configuration of the mono wing ornithopter (single set of wings). It consists of flapping-wing, horizontal stabilizer, vertical stabilizer, aileron, rudder, and elevator. Here the aileron is mounted on flapping-wing which is responsible for rolling moment of ornithopter. The rudder is used to control yaw moment and elevator is used to control pitching moment. The flapping of wing in upward direction is upstroke where as downward direction is down stroke. During the wing upstroke the lift distribution is smaller and more shifted towards the wing root. During the wing down stroke the lift distribution is bigger altogether than when gliding and more shifted towards the wing tip.

The flapping rate of the wing determines lift and forward propulsion of the unmanned aerial vehicle (UAV). The attitude of flapping-wing model can be controlled by elevator and rudder servos. The unsteady fluid dynamics of flapping-wings are poorly understood and it's difficult to get an ornithopter as desired[2]. The flapping-wing flight inherently produces very complex aerodynamic forces so that their motions are highly nonlinear and difficult to be modeled and controlled[3,4]. Since proper dynamics of the ornithopter are unavailable, so a platform is needed to test, simulate, and evaluate various control laws of the flapping-wing unmanned aerial vehicle. X-plane provides a facility to simulate flapping-wing ornithopter model.

Although there exist several simulators like Microsoft's flight simulator and flight gear, the X-plane was chosen since it provides extremely accurate aerodynamic and flight models also allows real time data to be sent in and out of the programme as well as airfoil design. X-plane has been used in the UAV research community as a visualization and validation tool for autonomous flight controllers[5,6]. The flight simulator (X-plane) provides very accurate aircraft models and has very

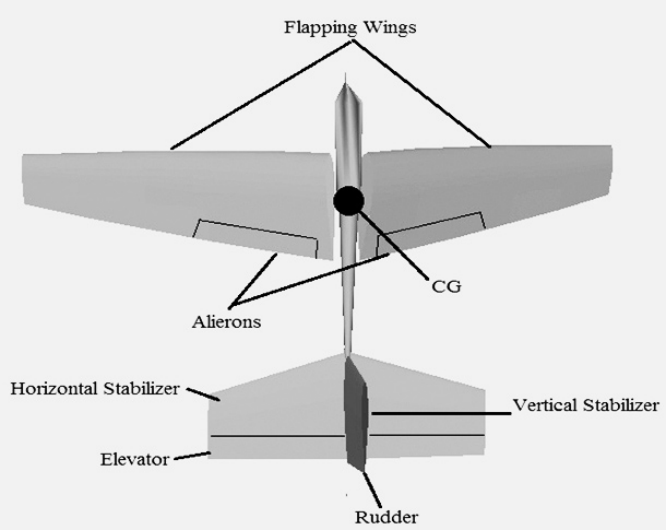


**Figure 1. Configuration of mono wing ornithopter.**

Received 03 August 2013, revised 19 March 2014, online published 21 July 2014

important feature: the possibility to exchange data with external systems[11]. Giving its realistic simulations capability, X-plane is also federal aviation administration (FAA) approved for pilot training. The aircraft models simulated in X-plane are built based on their exact physical dimensions and power among other characteristics of an engineering tool that can be used to predict the flying qualities of fixed and rotary wing aircraft[11].

Most other flight simulators use stability derivative methods to compute how an airplane flies. This technique involves simply forcing the aircraft nose to return to centered position along the flight path with certain acceleration for each degree of offset from straight-ahead flight of the airplane[9]. This is too simplistic to be used across the entire flight envelope of the airplane. Stability derivatives will not normally take into proper account, the asymmetric effects of engine failures, the chaotic effects of turbulence, stalls, spins and the myriad of dynamic effects which airplane generates. In other words, the commonly used stability derivatives are gross over simplifications of how an airplane flies. X-plane instead, assimilates the geometric shape of any aircraft and then figures out how an aircraft will fly. It does this by an engineering process called 'blade element theory', which involves breaking of aircraft down into many small elements and then finding forces on each little element several times per second. These forces are then converted into accelerations which are then integrated to velocities and positions. This method of computing the forces on the airplane is much more detailed, flexible and advanced than the flight model that is used by most other flight simulators. By doing this process, X-plane accurately predicts performance and handling qualities of an airplane of given geometry[11].

The main components of the test platforms are as follows:

- Matlab/Simulink containing the autopilot control system.
- Flight simulator X-plane containing the flapping-wing model to be controlled.
- Microcontroller to command the control surfaces (rudder, elevator and aileron).

The test platform concept proposed here is based on the block diagram presented in Fig. 2. In the test platform, the block controller is replaced by the designed autopilot system model and it runs into the Matlab/Simulink environment. Similarly, the block aircraft dynamics is replaced by the flight simulator X-plane with the flapping-wing model that is to be controlled. Thus, the basic principle of the test platform is establishing the communication between Matlab/Simulink, X-plane, and microcontroller. The parameters calculated by the autopilot

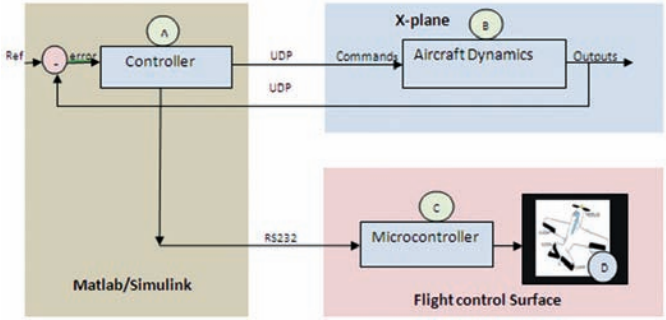


**Figure 2. Block diagram of proposed test platform.**

control system are sent to X-plane in order to command the ornithopter's control surfaces. The X-plane calculates the new attitude of ornithopter according to the inputs received from Matlab/Simulink. The X-plane sends those new ornithopter attitude parameters back to Matlab/Simulink closing the loop.

Matlab/Simulink restarts the process by providing updated commands to X-plane ornithopter model. The inputs given to the ornithopter control surfaces in the X-plane are simultaneously sent to a microcontroller which translates these commands from Matlab/Simulink into servo movement. The model aircraft flight control surfaces reproduce the same deflection observed on the X-plane aircraft. The communication between Matlab/Simulink and X-plane is made through user datagram protocol (UDP). Between Matlab/Simulink and microcontroller it uses RS-232 serial communication at a baud rate of about 9600 bps. Block diagram shown by Fig. 3 summarizes the three axis flapping-wing autopilot systems.

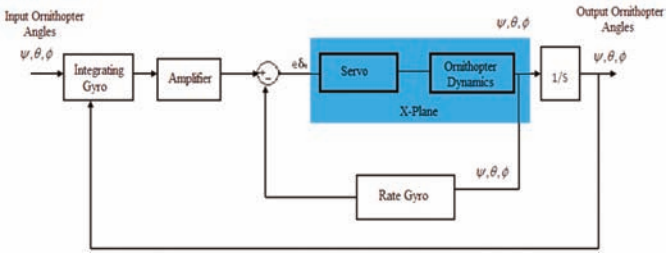


**Figure 3. Three axis flapping wing autopilot system.**

The microcontroller represented by block C on Fig. 2 symbolizes the experimenter board Arduino-ethernet shield. It is a development kit named Arduino-ethernet shield with a clock of 16 MHz, low power consumption, featured with a variety of I/O (digital, PWM), RAM and flash memory space. The choice for this device is justified by its in-built ethernet microcontroller chip, serial port (communicate between i/p and o/p devices in the external world through its Ethernet port) as well as easy integration with digital servos through PWM I/O ports. The data's from Matlab/Simulink serial port are received from microcontroller serial port pin 0 (RX), pin 1 (TX) and drives the servo by generating the PWM signal in pin 12, 13 and 14. It uses Arduino programming language to program controller in the Arduino development environment. On block D of Fig. 2 represents the digital servo that commands model aircraft flight control surfaces.

## 2. X-PLANE DATA INTERFACE

Flight simulator X-plane has an important feature that is essential to this test platform development. It has capacity of sending and receiving data from other devices. One way to implement this communication is by employing UDP. Data packets are sent and received through the computer ethernet port[6]. Figure 2 shows UDP being used to establish the communication between Matlab/Simulink (controller) and X-plane (aircraft dynamics).

UDP uses a simple transmission model without implicit hand-shaking dialogues for guaranteeing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and data packets may arrive out of order or appear duplicated or

may be missed without notice. Error checking and correction are not considered for the performance of this application. This way, UDP avoids the overhead of such processing at the network interface level being extremely fast[7]. Hence UDP speed constitutes a key point in the test platform, once the communication between Matlab/Simulink and X-plane is established. It must be fast enough to synchronize commands, process data and response to system. X-plane is able to send or receive up to 99.9 data packets per second via UDP. Each data package may be configured to carry aircraft parameters data that are selected on X-plane data input and output interface. For example, in the case of lateral motion, parameters such as roll, yaw, pitch, altitude and speed shall be selected for transmission. Each parameter receives a numeric label for proper identification.

## 3. PLANE MAKER SETUP

Plane maker is a tool in X-plane software package which help users to design their custom build aircraft. Once all the physical specifications of the airplane have been entered (e.g., weight, wing span, control deflections, engine power, airfoil sections, etc.), the X-plane simulator will predict how the aircraft flies[10]. The flapping properties of wings in plane maker environment can be created by varying its flapping amplitude (i.e. maximum and minimum dihedral angle of the flapping-wing model). It can be set by selecting expert tab – airfoils – wings – variable-dihedral. The maximum and minimum flapping angle of ornithopter may be set by using maximum dihedral tab. Once the model has been created in the Plane maker environment; it creates an .acf format file. The .acf file is ready to run the created flapping-wing ornithopter model in the X-plane environment.

## 4. X-PLANE SETUP

X-plane sends data in the form of different sentences, each sentences are divided into 41 bytes of data. The first four bytes of the packet represents the characters DATA used to indicate that, this is a data packet. The fifth byte is an internal code (I). The next four bytes represent the parameter label (L1, L2, L3, and L4). Following 32 bytes (B11, B12, B13, B14 to B81, B82, B83, B84) represents the data itself in single precision floating point[11]. Taking each of 4 bytes, the first bit is the sign bit. It tells whether the number is positive or negative. The next 8 bits are biased exponent, remaining 23 bits represents the mantissa. The rest of the bytes complete the data. So, the Matlab/Simulink model has to decode this data packet accordingly[6,8].

## 5. MATLAB/SIMULINK SETUP

The instrumentation and control toolbox in Matlab/Simulink can play a major role in sending and receiving data from X-plane. The UDP receive block in Matlab/Simulink environment will receive data from the IP address where the X-plane software is to be run. After receiving the data from the IP address, it unpacks data into an array of bytes. The received data are in the form of sentence, after decoding it, we can get corresponding decimal data like throttle, wing incident, pitch, roll and yaw of the flapping-wing unmanned aerial vehicle. Here the throttle represents the flapping rate of wings which varies its flapping amplitude. Generally the ornithopter wing is attached to the body at a slight angle called angle of attack. The down stroke of the wing deflects air both downward and backward forces generating lift and forward thrust.

Sending array of bytes from Matlab/Simulink to X-plane is similar to that of data received from X-plane. While sending the data to X-plane make sure that the fifth byte is set to zero. The remaining bytes are used to denote the index number and to send the input commands[7] (maximum eight input commands transmitted per sentence). Figure 4 shows complete closed loop implementation of flapping-wing ornithopter. To receive data in the X-plane software, we must specify the IP-address of the receiver (Matlab/Simulink) system which is available in the net connection advanced tab. Once the parameters are selected on X-plane, through UDP it is possible to send and receive data from Matlab/Simulink environment. Figure 5 shows various result of ornithopter executed in the X-plane environment.

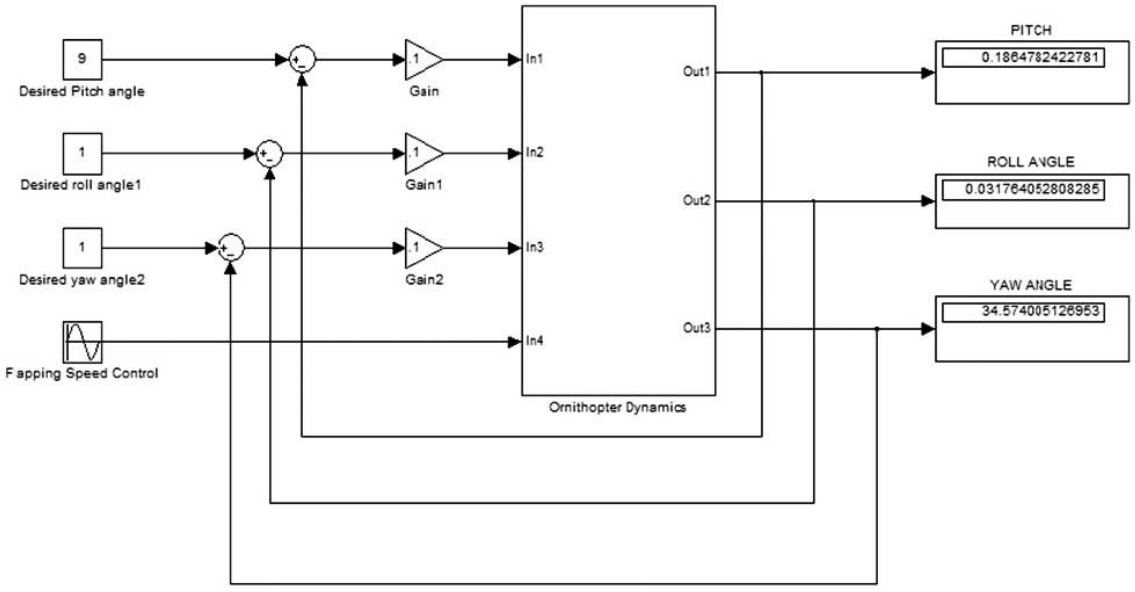


**Figure 4. Closed loop implementation of flapping-wing ornithopter.**

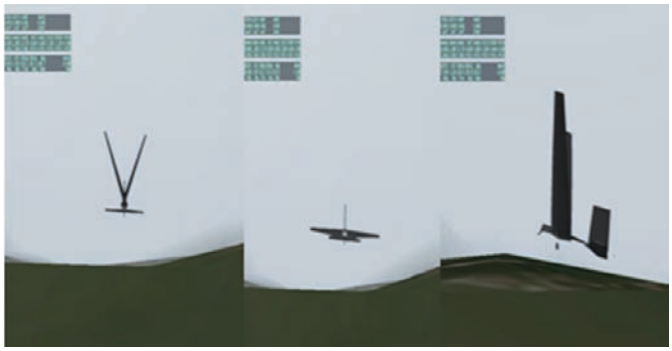

**Figure 5. Results of flapping-wing model in X-plane.**

## 6. OBSERVATION AND RESULT

Figure 6 shows the proposed simulation platform setup of the pilot control panel developed using Matlab/Simulink to control X–plane flapping-wing model in simulation mode. To initialize the simulation, the X-plane is loaded with designed flapping-wing model at 100 ft altitude. As soon as the test platform runs in pilot control panel, the designed autopilot system will take flapping-wings attitude control. During simulation it is possible to observe that the change in 20 degree input pitch angle will reflect in the designed flapping-wing pitch angle and performs a slight pitch up in the X-plane simulator. Similarly roll and yaw performances are also observed in the simulation environment. It is Interesting to notice that even with the reference inputs, there are small commands to the control surfaces. This is something that was only possible to observe due to the realism provided by X-plane simulation which introduces small perturbations to aircraft flight reproducing a real atmosphere with wind, turbulences, etc. One exercise that can also be performed on the designed autopilot system is to vary the loop gains and check out the flapping-wing platform responses. This can be used to tune the loop gains and to optimize system performance. In future, the controller part in Fig. 2 is replaced by external hardware with embedded control law algorithm. So the system can be developed into a real-time prototype model.

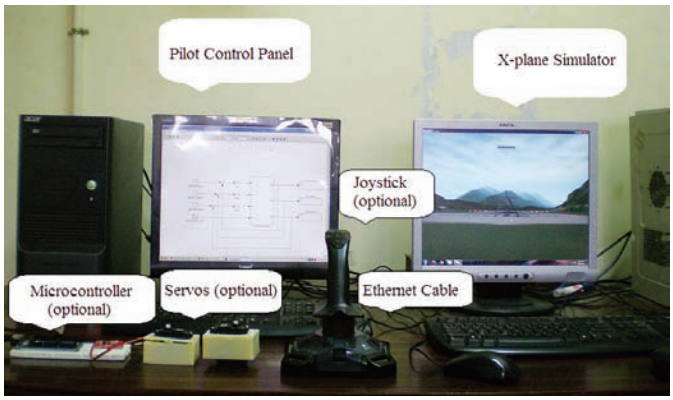


**Figure 6. Proposed simulation platform setup.**

## 7. CONCLUSION

In this paper, the design and making of flapping-wing simulating model using plane maker software was discussed. Interface between MATLAB/SIMULINK and X-plane software was discussed in detail. The development of this test platform resulted in a valuable tool for the students, researchers to aid autopilot system study and to design flapping-wing model. It allows monitoring the responses of a designed autopilot system for the flapping-wing unmanned aerial vehicles. The responses of developed flapping-wing unmanned aerial vehicle which was verified with respect to the above designed model and the suitable PID Gains for it has been found. In future, an autopilot (hardware) may interface in the loop to make the system further realistic.

## ACKNOWLEDGEMENTS

The author would like to thank all the faculty and students in Division of Avionics, Madras Institute of Technology, Chennai for their encouragement and continuous support for the research.

## REFERENCES

1. Baek, S.S.; Bermudez, Garcia L.F & Fearing, S.R. Flight control for target seeking by 13 gram ornithopter. *In* proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, September 2011.
2. Zachary, J.J. Design and construction of an autonomous ornithopter, Department of Mechanical Engineering, Massachusetts Institute of technology, Cambridge, United States, 2009.
3. Han, Jae-Hung; Lee, Jin-Young & Dae-Kwan Kim. Ornithopter modeling for flight simulation. *In* proceedings of the International Conference on Control, Automation and Systems, Korea, October 2008.
4. Park, Joon Hyuk & Yoon, Kwang-Joon. Designing a biomimetic ornithopter capable of sustained and controlled flight. *J. Bionic. Eng.*, 2008, **5**(1), 39-47.
5. Ratti, Jayant & Vachtsevanos, George. Inventing a biologically inspired, energy efficient micro aerial vehicle. *J. Intell. Robot Syst*., 2012, **65**(1-4), 437-455.
6. Sérgio Ronaldo Barros dos Santos, Sidney Nascimento Givigi Junior, Cairo Lúcio Nascimento Júnior, & Neusa Maria Franco de Oliveira. Modeling of a hardware-in-the-loop simulator for UAV autopilot controllers. *In* proceedings of 21st Brazilian Conference on Mechanical Engineering, Brazil, October 2011.
7. Ernst, Daniel. Development of research platform for unmanned vehicle controller design, evaluation, and implementation system: From MATLAB to hardware based embedded system. Department of Computer Science and Engineering University of South Florida, Florida, 2007. PhD Thesis.
8. Adiprawita, W.; Ahmad, Adang suwandi & Sembiring, Jaka. Hardware in the loop simulation for simple low cost autonomous UAV (Unmanned Aerial Vehicle) autopilot system research and development. *In* proceedings of the International conference on Electrical Engineering and Informatics, Bandung Institute of Technology, Indonesia, June 2007.
9. Iain, McManus & Rodney, Walker. Simulation for the next generation of civilian airspace integrated UAV platforms. *In* the proceedings of AIAA Modeling and Simulation

Technologies Conference and Exhibit, Providence, Rhode Island, August 2004.
10. Plane Maker for X-Plane 10. Laminar Research, 2013.
11. X-plane 10. Laminar Research, 2013.
12. Instrumentation Control Toolbox for MATLAB and Simulink. http://www.mathworks.com.

## CONTRIBUTORS

**Mr A. Kaviyarasu** received his BE (ECE) and ME (Avionics) from Anna University, Chennai. Currently he is working as a Teaching fellow in the Department of Aerospace Engineering, MIT campus, Anna University, Chennai. He has been awarded as 'Excellence in Innovation' by Anna University for his contribution in the field of unmanned aerial vehicle. His area of interest are : UAV modeling and simulation and satellite communication.

**Dr K. Senthil Kumar** received his ME (Avionics) and PhD (Avionics) from Anna University (MIT), Chennai. Currently he is working as an Associate Professor in the Department of Aerospace Engineering, MIT campus, Anna University, Chennai. He has been honored by various awards for his contribution in the field of unmanned aerial vehicle. He is member of AIAA, ASI and system society of India. His area of interests are : Unmanned aerial vehicles, avionics system and aerospace guidance and control.