Chapter 30
Design, Development and Evaluation of Longitudinal Autopilot for An Unmanned Aerial Vehicle Using X-Plane/Simulink

A. Kaviyarasu, P. Sivaprakash and K. Senthilkumar

Abstract Presently there is a vast interest in unmanned aerial vehicle (UAV) development given its civilian and military applications. One of the main UAV components is autopilot system. Its development invariably demands several lab simulations and field tests. Generally after an UAV crash few parts remain unused. Thus, before embedding an autopilot system, it has to be exhaustively lab tested. With educational and research purposes in autopilot control systems development area, a test platform is herein proposed. It employs Matlab/Simulink to run the autopilot controller under test. The autopilot controller designed on Matlab/Simulink is tested by controlling an aircraft on X-Plane. The inputs given to the aircraft flight control surfaces in the X-Plane are simultaneously sent to the microcontroller which translates these commands into effective servo movement control. Through this platform, designed autopilot systems can be applied into models similar to real aircraft minimizing risks and increasing flexibility for design changes. As study case, tests results from a pitch attitude autopilot system are presented.

Keywords Autopilot · X-plane

A. Kaviyarasu (✉) · P. Sivaprakash · K. Senthilkumar
Department of Aerospace Engineering MIT, Anna University, Chennai, India
e-mail: isrokavi@gmail.com

P. Sivaprakash
e-mail: psivaprak07@gmail.com

R. Malathi and J. Krishnan (eds.), *Recent Advancements in System Modelling Applications*, Lecture Notes in Electrical Engineering 188,
DOI: 10.1007/978-81-322-1035-1_30,

30.1 Introduction

The advancement of modern aircraft design requires the development of many technologies such as aerodynamics, structures, materials, propulsion and flight controls. Currently, the aircraft design strongly relies on automatic control system to monitor and control many of the aircraft subsystems. Those control systems can also provide artificial stability to improve the flying qualities of an aircraft.

To stabilize the aircraft after being disturbed from its wing-level equilibrium flight attitude, the autopilot system was primarily conceived [1]. The modern autopilot systems are much more complex and are essential to flight aiding crew in navigation, flight management, stability augmentation and landing operations.

With the development of modern aircraft new problems arose due to their dynamic stability. For instance, some oscillations are inherent to the aircraft can be adequately damped or controlled by the pilot if the period is around 10 s or more. On the other hand, if the period of oscillation is 4 s or less, the pilot's reaction time is very short. Thus, such oscillations have to be artificially well damped [2].

The oscillations named "Short period" in pitch for longitudinal motion, and "Dutch roll" for lateral-directional motion fall into the category of a 4 s oscillation. In more conventional aircraft that kind of oscillation may be damped by handling their constructive characteristics. However, for modern jets artificial damping is required. It is provided by an automatic system [2].

The above concepts are also applicable to unmanned aerial vehicle (UAVs) whose development are presently given a vast interest in increasing possibilities of its civilian and military applications. Some of the applications are reconnaissance, surveillance, search and rescue, remote sensing, traffic monitoring missions, etc. [3].

One of the main components of UAVs is the autopilot system. An autopilot is a mechanical, electrical, or hydraulic system used to guide vehicle without any assistance from human being. The autopilot systems development invariably demands several lab simulations and field tests. The later ones are of high risk particularly when conducted on UAVs. Generally after an UAV crash few parts remain usable. Thus, before embedding an autopilot system, it has to be exhaustively lab tested.

Based on this necessity, with educational and research purpose in autopilot control systems development area, a test platform is here in proposed and various control laws are verified before making it to the final integration process.

30.2 Objective

This article intends to describe a test platform developed to aid in autopilot design process.

The test platform herein proposed provides an environment where the designed autopilot system can be applied into models similar to real aircraft. Several flight

situations can be simulated. Parameters of flight as well as aircraft responses can be monitored and easily analyzed. These features increase the flexibility to implement changes and immediately check out the results.

Basically, this test platform employs the Matlab/Simulink to run the autopilot controller to be tested, the flight simulator X-plane with the aircraft to be commanded, a microcontroller to command model aircraft flight control surfaces and a servo to drive these control surfaces. All this resources are interconnected through data buses in order to exchange information.

As study case, this article presents the results obtained from tests conducted over an autopilot control system designed for longitudinal movement, specifically for pitch mode control. This system was conceived for Cessna aircraft.

30.3 Aircraft Translational Motions

An airplane in free flight has three translational motions (vertical, horizontal and transverse), three rotational motions (pitch, yaw and roll) and numerous elastic degrees of freedom.

In order to reduce the complexity of this mathematical modeling problem, some simplifying assumptions may be applied. First, it is assumed that the aircraft motion consists of small deviations from its equilibrium flight condition. Second, it is assumed that the motion of the airplane can be analyzed by separating the equations into two groups [2]:

- Longitudinal Equations: composed by X-force, Z-force and pitching moment equations.
- Lateral Equations: composed by Y-force, yawing and rolling moment's equations.

Figure 30.1 shows the above mentioned aircraft reference axes, forces and moments.

30.3.1 Longitudinal: Directional Motion

The longitudinal-directional motion of an aircraft disturbed from its equilibrium state is pitching.

The longitudinal-directional motion, specifically the pitch motion, is the movement intended to be simulated on the proposed test platform as demonstration example. To simplify the studied case, it was chosen to address specifically the pitch motion in this work. Even though this simplifies the dynamic model equations, the platform functionality and characteristics, are very well demonstrated.

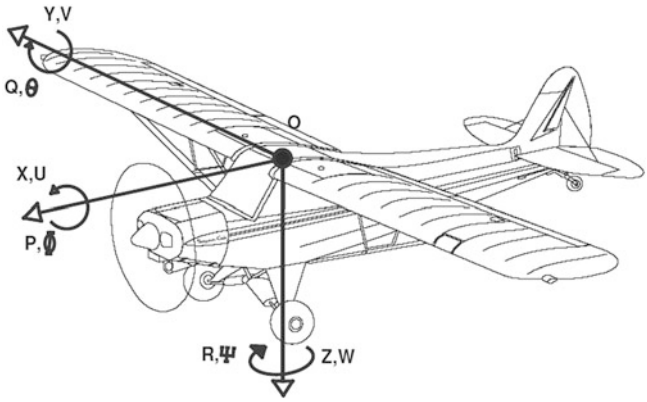

Fig. 30.1 Aircraft axes, forces and moments

30.3.2 Pitch Motion Closed-Loop Control System

The aircraft pitch motion can be controlled by an autopilot system block diagram as shown in the Fig. 30.2.

Through this simplified system, the pitch angle (θ) can be controlled by a reference angle applied as an input. Primarily, the pitch attitude autopilot is designed to maintain the aircraft with leveled wings, or with $\theta = 0$. Usually, this simplified system does not fulfill the control design performance requirements in relation to damping ratio, overshoot and undamped natural frequency.

So that, in the case study herein presented, a more efficient control system is employed. It is shown in Fig. 30.3.

In this case, an accurate aircraft pitch to elevator transfer function for Cessna 172SP aircraft is employed in the pitch autopilot system. This transfer function was calculated based on aircraft physical characteristics and stability derivatives as detailed in [4]. Moreover, a typical elevator actuator transfer function is employed.

The pitch attitude autopilot system gains *KG, Ka* and *KRG* are calculated through the root locus method considering the above rationale.

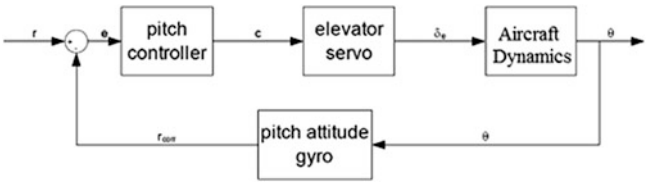

Fig. 30.2 A pitch attitude autopilot system

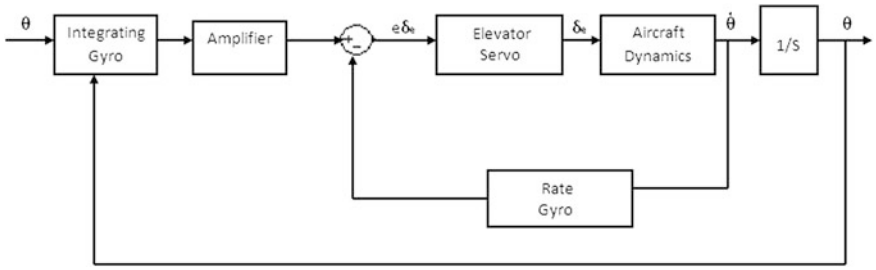

Fig. 30.3 Pitch attitude autopilot system

30.4 The Test Platform

The main components of the test platforms are as follows.

- Matlab/Simulink containing the autopilot control system.
- Flight Simulator X-Plane containing the aircraft model is to be controlled.
- Microcontroller to command the aircraft flight control surfaces.

The test platform concept is based on the block diagram presented in Fig. 30.4. In the test platform, the block "Controller" is replaced by the designed autopilot system model. This model runs into the Matlab/Simulink environment.

Similarly, the block "Aircraft Dynamics" is replaced by the flight simulator X-Plane with the aircraft model that is to be controlled.

Thus, the basic principle of the test platform consists of establishing the communication between Matlab/Simulink, X-Plane, microcontroller and servo in the following mode: The parameters calculated by the autopilot control system are sent to X-Plane in order to command the aircraft flight control surfaces. The X-Plane calculates the new aircraft attitude according to the inputs received from Matlab/Simulink. The X-Plane sends those new aircraft attitude parameters back

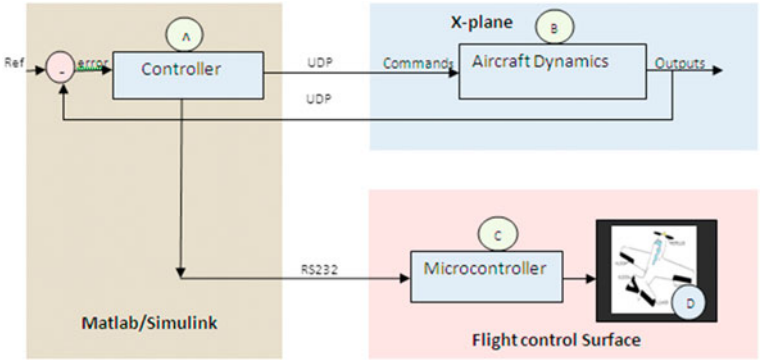

Fig. 30.4 Test platform block diagram

to Matlab/Simulink closing the loop. Matlab/Simulink restarts the process by providing updated commands to X-Plane aircraft. The inputs given to the aircraft flight control surfaces in the X-Plane are simultaneously sent to microcontroller which translates these commands received from Matlab/Simulink into effective servo movement control. The microcontroller calculates the deflection angle to be imposed to the flight control surfaces and generates a proportional PWM signal to command the servo. 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 UDP (User Datagram Protocol). Between Matlab/Simulink and microcontroller a RS-232 serial communication is used. Block diagram shown by Fig. 30.4 summarizes the platform concept.

The flight simulator X-Plane provides very accurate aircraft models and has a very important feature. That is the possibility to exchange data with external systems [5]. 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 real physical dimensions, power and weight among other characteristics. X-Plane is not considered a game, but can be categorized as an engineering tool that can be used to predict the flying qualities of fixed and rotary wing aircraft [5].

Most other flight simulators use stability derivatives method to compute how an airplane flies. This technique involves forcing the aircraft nose to return to a centered position along the flight path with certain acceleration for each degree of offset from straight-ahead flight of the airplane. 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 that airplane flight generates. In other words, the commonly used stability derivatives are gross over simplifications of how an airplane flies. In summary, those simulators can not predict how the airplane will fly. Basically, the airplane designer teaches the simulator how the airplane should fly, and the simulator reproduces that information right back to the user [5].

X-Plane instead, assimilates the geometric shape of any aircraft and then figures out how that aircraft will fly. It does this by an engineering process called “blade element theory”, which involves breaking the aircraft model down into many small elements and then finding the forces on each little element many 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 what will be the performance and handling qualities of an airplane of given geometry [5]. The microcontroller represented by block C on Fig. 30.4 symbolizes the experimenter board Arduino-Ethernet shield as shown in Fig. 30.5.

This is a development kit based Arduino Ethernet Shield with a clock of 16 MHz, low power consumption, featured with a variety of I/O (digital, PWM), besides RAM and flash memory space. The choice for this device is justified by its

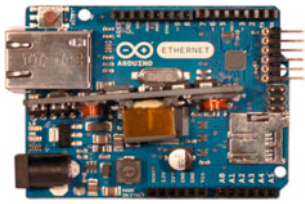

Fig. 30.5 Microcontroller unit

Fig. 30.6 Model aircraft digital servo

inbuilt Ethernet microcontroller chip, serial port (communicate between I/O devices in the external world) as well as easy integration with digital servos through PWM I/O ports. On block D of Fig. 30.4 it represents the digital servo that commands model aircraft flight control surfaces. A typical servo is shown in Fig. 30.6.

It receives PWM signals from the microcontroller unit and converts them into proportional axis movement.

30.5 X-Plane Data Interface

Flight Simulator X-Plane has an important feature that is essential to this test platform development. Its capacity of sending and receiving data to and from X-plane software and MATLAB. One way to implement this communication is by employing protocol UDP. Data packets are sent and received through the computer Ethernet port. Figure 30.4 shows UDP protocol 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, appear duplicated or go missing without any notice. Error checking and correction are considered either not necessary or performed in the application. This way, UDP avoids the overhead of such processing at the network interface level being extremely fast [6]. Time-sensitive applications such as this test platform often use UDP. Dropping packets

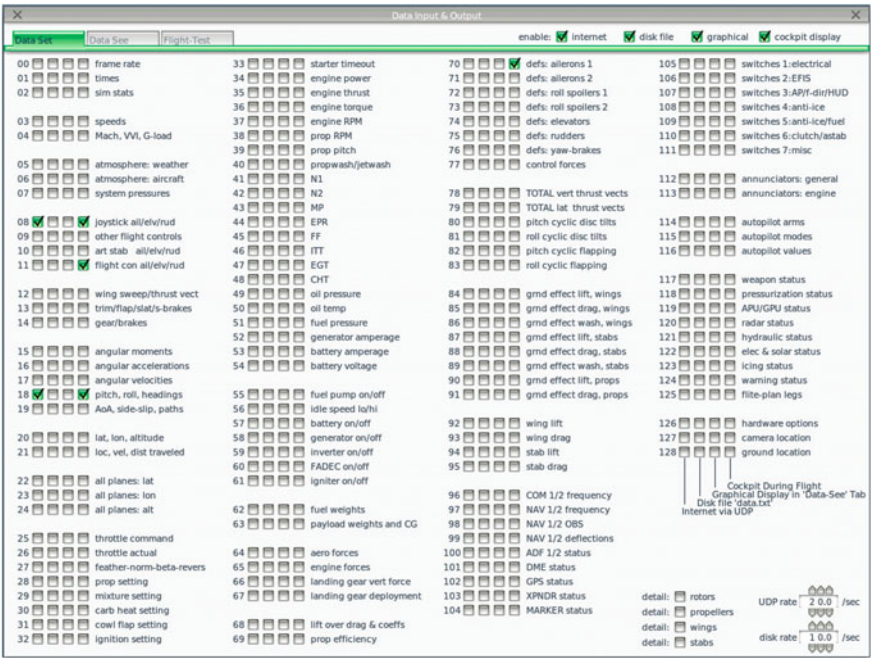

Fig. 30.7 X-plane parameters user interface

is preferable to waiting for delayed packets, which it is not an option in real-time systems.

Hence UDP speed constitutes a key point in the test platform, once the communication between Matlab/Simulink and X-Plane must be fast enough to synchronize commands, data processing and system responses.

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.

Figure 30.7 shows the interface that serves to select data for input and output from X-Plane.

Besides parameters selection for UDP data transmission or receiving, it also serves for selecting parameters to be shown during aircraft flight simulation, file recording and graphs construction.

Once the parameters are selected on X-Plane, through UDP it is possible to receive them at Matlab/Simulink environment and after processing, to send commands back to X-Plane aircraft. In the case of pitch attitude autopilot proposed example, Matlab/Simulink generates the pitch angle reference and control signal sending them to X-Plane. In its turn, X-Plane commands the simulated aircraft

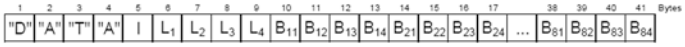

Fig. 30.8 X-plane data package pattern

according to the inputs received. The new aircraft pitch angle position is sent from X-Plane back to Matlab/Simulink, restarting the process.

The X-Plane data package follows a pattern shown on Fig. 30.8.

Basically it is composed by a sequence of bytes that need to be properly interpreted. X-Plane uses what are known as single precision floating point variables for just about everything sent over the network. This means that the numbers can be stored using four bytes. The first four bytes of the packet shown on Fig. 30.7 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 there are eight sets of four bytes (B11, B12, B13, B14 to B81, B82, B83, B84) representing the data itself in single precision floating point. Taking each set of four bytes, the first bit is the sign bit. It tells whether the number is positive or negative. The next eight bits are the biased exponent. The remaining 23 bits represents the mantissa. The next seven sets of four bytes completes the data. So, the Matlab/Simulink model has to decode this data packet accordingly. It has to properly encode the data to be sent to X-Plane. Most of the blocks of the test platform implemented on Matlab/Simulink environment are dedicated to decode and encode data packets from and to X-Plane.

30.6 Implementation

Figure 30.9 illustrates the test platform implemented.

In one computer Matlab/Simulink runs a model containing the autopilot control system and other blocks responsible for UDP data packets decode and encode process as well as serial communication with microcontroller. In the other computer X-Plane simulates the aircraft to be controlled. Both computers communicate to each other through their Ethernet port using UDP protocol. Through serial bus the microcontroller receives the same commands Matlab/Simulink sent to X-Plane. It decodes this data converting them into PWM signal. An interface amplifies this signal to apply it to the servo that commands the model aircraft flight control surfaces.

30.7 Application Example: Test Results

In order to demonstrate how the test platform could be applied in laboratory classes and also help autopilot systems development, the roll attitude autopilot system was designed and integrated into the platform. The dynamic system used as

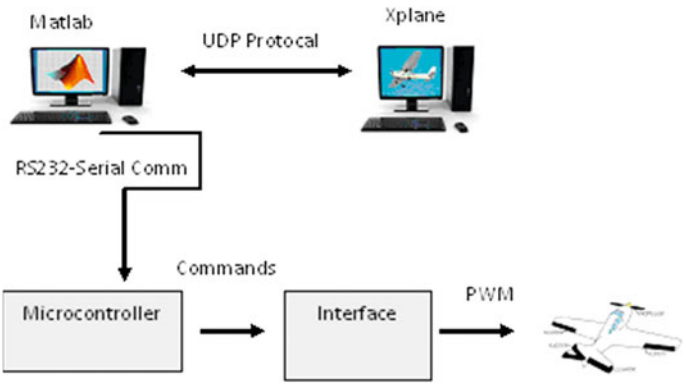

Fig. 30.9 Autopilot test platform implementation

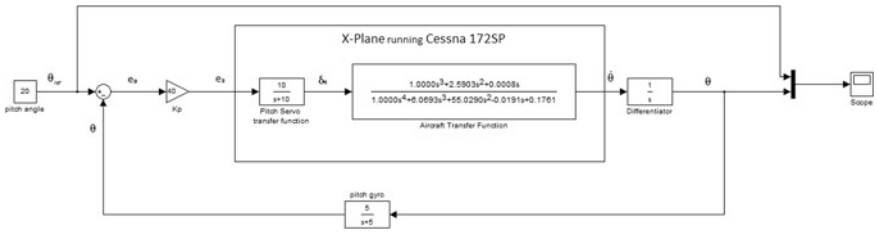

Fig. 30.10 Pitch attitude autopilot system for Cessna 172SP

the case to be studied and tested using the proposed platform can be represented by Fig. 30.3. The gains *KG, Ka* and *KRG* of that system were calculated through the root locus method using the systems dynamics as in (2) and (3), which are the Cessna pitch to elevator and typical Pitch servo transfer functions.

Pitch Dynamics:

$$\frac{N_{\delta_e}^{\theta}}{\Delta_{long}} = \frac{1.0000s^3 + 2.5903s^2 + 0.0008s}{1.0000s^4 + 6.0693s^3 + 55.0290s^2 - 0.0191s + 0.1761}$$

Pitch Servo:

$$S(s) = \frac{10}{s + 10}$$

The designed system result is shown on Fig. 30.10. It is also shown which blocks will be simulated by X-Plane and which ones will be integrated into the test platform model at Matlab/Simulink.

The Pitch to elevator and servo transfer functions are simulated by the Cessna 172 SP at X-Plane software. The other autopilot blocks are distributed into the test

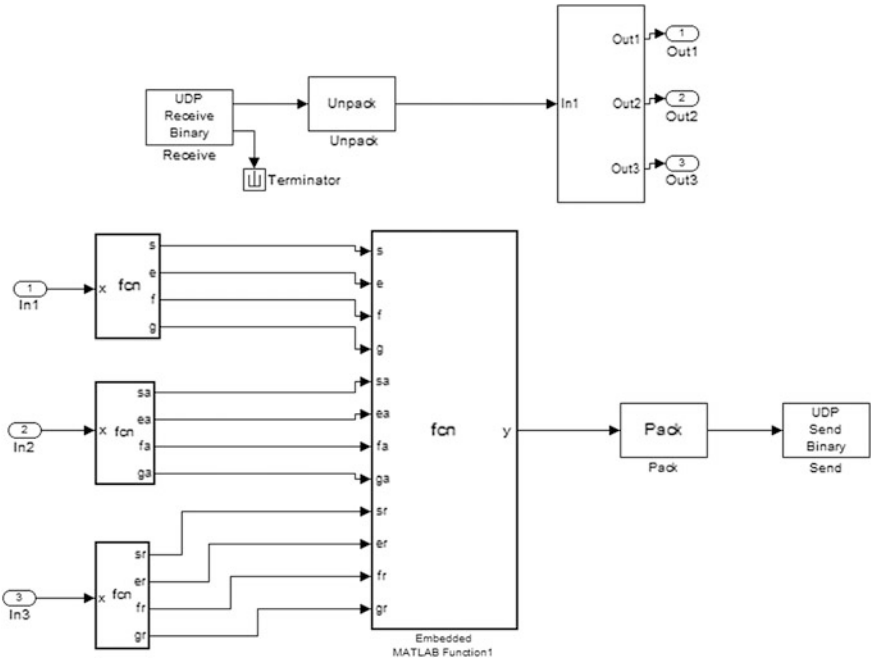

Fig. 30.11 Autopilot test platform model at matlab/simulink

platform model. Figure 30.11 shows the entire test platform model implemented at Matlab/Simulink environment.

In a typical lab class, the student should find the test platform ready to be used with the communication interfaces appropriately configured. The gains *KG, Ka* and *KRG* shall then be fed into the controller implemented at Matlab/Simulink. The gains calculation may be performed at the lab or can be previously done, being the lab classes preferably used for test at the platform.

To initialize the simulation, X-Plane is loaded with aircraft Cessna 172 SP in a cruise flight at 40,000 ft altitude. As soon as the test platform model shown on Fig. 30.11 runs, the designed autopilot system takes the aircraft control. In principle, for any pitch angle applied as reference into the autopilot system, the aircraft at X-Plane shall respond following that reference.

The reference signals applied as system input, the system response and also the commands sent to the aircraft flight control surface actuators can be monitored through real time graphs at Matlab/Simulink. In the proposed study case, the input is the reference pitch angle, the system response is the new aircraft Pitch angle attitude and the commands are the deflections to be imposed to the elevators. Figure 30.12 shows the results for the proposed pitch attitude autopilot system implemented at the test platform.

The graph on Fig. 30.12 presents the autopilot Pitch angle reference 20° applied into the system. During simulation it is possible to see the Cessna 172SP make a

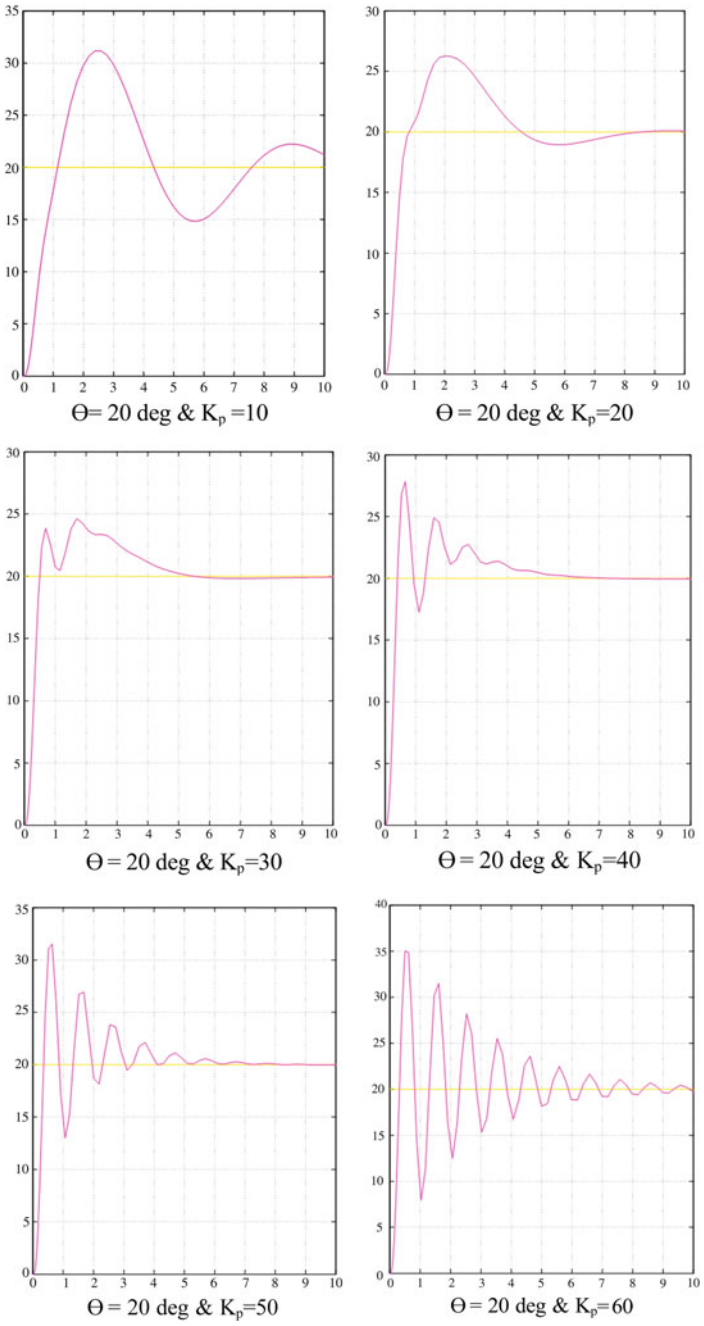

Fig. 30.12 Pitch reference and response

20° pitch angle attitude and performing a slight pitch up. In parallel it is possible to verify the servo movements in response to these commands in the aircraft model. Interesting to notice that even with the aircraft leveled and keeping a certain pitch attitude there are small commands to the flight control surfaces. This is something that only was 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 aircraft responses. This can be used to properly tune the loop gains to optimize system performance.

30.8 Conclusion

The development of this test platform resulted in a valuable tool for the students and professionals to aid autopilot system study and design. It allows monitoring the aircraft responses for a designed autopilot system with high degree of realism. It is possible to change control system parameters very easily and check the results out in a friendly environment. This possibility easies the design task as well minimizes risks of embedding the system for field tests.

The test platform also permits the study of longitudinal and lateral-directional movement of different fixed and rotary wing unmanned aerial vehicle platform for which the autopilot system is being designed. A possible extension of this study is to test the platform autopilot design with the Hardware in loop system (HILS) to simulate the UAV in the Real-time.

References

1. Nelson RC (1998) Flight stability and automatic control, 2nd edn. McGraw-Hill, New York
2. Blacklock JH (1991) Automatic control of aircraft and missiles. Wiley Inter science Publication, New York
3. Suwandi Ahmad A, Sembiring J (2007) Hardware in the loop simulation for simple low cost autonomous UAV (unmanned aerial vehicle) autopilot system research and development. Institute Technology Bandung, Indonesia
4. Laminar research (2009) X-plane description, X-plane manual
5. AeroSim-Aeronautical Simulation block set (2005) Version 1.1, User's Guide, unmanned dynamics, USA
6. Roskam J (1982) Airplane flight dynamics and automatic flight controls, part I roskam aviation and engineering corporation