International Journal of Engineering & Technology, 7 (4.25) (2018) 53-61

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Numerical Simulation Platform for a Generic Aircraft Flight Dynamic Simulation

Adam Adila Arif[1]*, Stepen[2], Rianto Adhy Sasongko[3]

Institut Teknologi Bandung, Faculty of Mechanical and Aerospace Engineering, Indonesia
**Corresponding author E-mail: adamadilaarif@s.itb.ac.id*

Abstract

Flight simulation has been used for many purposes in several decades, started from pilot training, engineering purposes, even just for a hobby. In its development, computing technology has a vital role in aircraft simulation. Nowadays, people do an aircraft simulation mostly by using a set of the computing device and complementary equipment such as a joystick. On the other hand, engineers use aircraft simulation in order to evaluate the aircraft in many aspects such as flight performance, handling quality, and controllability. This paper proposes a simulation system to do an engineering simulation especially to evaluate controllability of an aircraft utilizing X-Plane 10, Python and MATLAB. This system can manipulate aircraft loading configuration and failure on aircraft systems, set the environment condition, and observe control-related flight parameters in MATLAB whilst the aircraft is still flying in the X-Plane 10 at the same time. The system had been tested and proved could work well with X-Plane 10.

Keywords: *Matlab, Simulink, X-Plane 10, Python, Flight Dynamics, Control, Engineering Simulator.*

1. Introduction

First aircraft simulator was built around 1910 named The Sanders Trainer which comprised a cockpit that could move when exposed to a fairly strong wind corresponding to the input of the pilot. Similar devices were developed by Walters and Antoinette in the same year, where cockpit motion was controlled by instructors [1]. Along with the development of analog computing technology in 1945-1965, modeling aircraft motion as sets of non-linear differential equations was easier. It also made a significant improvement in the fidelity of flight models and real-time simulation purpose as well [1]. Unlike analog computing, digital computing development was quite slow at that time until late 1970 when the development of transistor initiated major advance in digital computation as well, in this era processors could achieve an update rate around 50-60 Hz and visual systems began to be developed as well [1].
As microelectronics revolution began in 1980, standard PC outperformed minicomputers of 1970. For flight simulation, attention focused on the development of visual systems and smooth motion system as well [1].
Therefore, as a result of excellent computing technology, flight simulation software is now available for everyone and widely used for many purposes such as a hobby, cockpit familiarization, design validation, etc. Several examples of that software are, Microsoft Flight Simulator, X-Plane, Prepar3D, Flightgear, etc. X-Plane is a flight simulator produced by Laminar Research packaged with several commercial, military, and other type of aircraft as well as basic global scenery [4]. X-Plane also support Python-language based script with the help from Sandy Barbour's XPlugin Software Development Kit (SDK) called Python Interface [3]. This paper focused on usage of X-plane for aircraft simulation tool, especially in control aspect.
Aircraft control aspects are regulated in many documents, one of them is MIL-F-8785C that is discussed flying and handling qualities requirements in flight and on the ground mostly for manned, piloted aircraft. This document regulates the value of a stability-related parameter, such as damping ratio, Euler angle rate of change, etc, depending on stability modes of aircraft and configuration of the aircraft at that time [8]. Using that document, engineers validate their design whether it meets the requirements or not. In order to do that, engineers can perform a flight test using the real aircraft with real sensors, or it could be done in another way by utilizing aircraft simulator with a system to manipulate aircraft configuration and surrounding environment to examine aircraft dynamic characteristics and its control surface input as well.
In Institut Teknologi Bandung, a generic aircraft simulator is being developed in Dynamic Systems Laboratory, this simulator focused on numerical simulation platform, the figure below shows development phase of this simulator.
There is two type of simulator that used in our laboratory as shown in Figure 1 that cannot be executed at the same time, if we use the first one, aircraft dynamic model in X-Plane will be freezed because we want to simulate another aircraft modeled in Matlab and Simulink. And if we want to use the second one, we use X-Plane in normal mode without interruption from Matlab and/or Simulink. First type of simulator is using built-in X-Plane dynamic model and the second one is using a dynamic model which is built in Simulink and Matlab with aircraft aerodynamic database obtained from DATCOM+, here X-Plane is used for visualization purpose only. Further detail is discussed in next chapter.

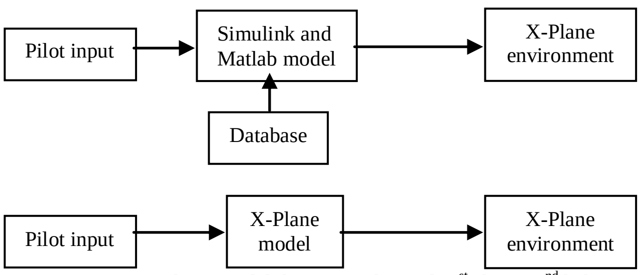


Fig. 1: Two simulator model that currently used (1st above, 2nd below)

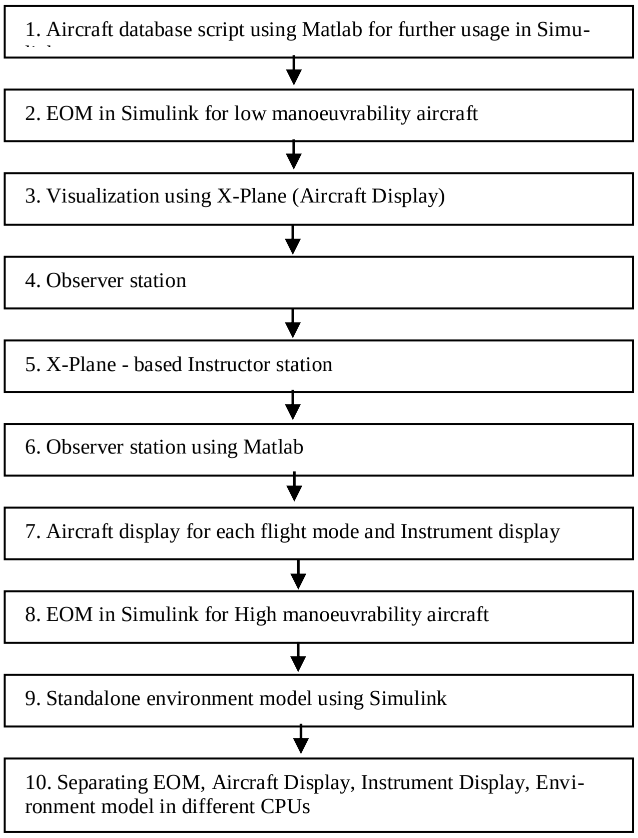

1. Aircraft database script using Matlab for further usage in Simu-
2. EOM in Simulink for low manoeuvrability aircraft
3. Visualization using X-Plane (Aircraft Display)
4. Observer station
5. X-Plane - based Instructor station
6. Observer station using Matlab
7. Aircraft display for each flight mode and Instrument display
8. EOM in Simulink for High manoeuvrability aircraft
9. Standalone environment model using Simulink
10. Separating EOM, Aircraft Display, Instrument Display, Environment model in different CPUs

Fig. 2: Development phase of Generic Aircraft Simulator

This paper will briefly explain what generic aircraft simulator is, how it works, and how far we develop this simulator at this moment for the purpose of aircraft control aspects observation in section 2, 3 and 4. Conclusion and suggestion based on the work that we have done are presented in section 5.

2. Generic Simulator Architecture

Generic means not specific, so we aimed that our simulator is suitable for several kinds of aircraft. Simulator development was began in early 2000s. Firstly by building script that automatically extract output data from DATCOM+. After that we built equations of motion block in Simulink to accommodate pilot input using joystick for low maneuverability aircraft class. To give visual cue to pilot for observing aircraft motion we built python-based script to visualize aircraft motion in X-Plane. Lastly, we create an aircraft observer and pilot instructor using Matlab, X-Plane, and python script, as shown in Figure 2.

There are some objectives that we made by constructing this simulator:
- As a tool for analyzing aircraft dynamic characteristics.
- As a tool for aircraft control implementation.
- As a tool for validating the aircraft design.

Therefore, the components that should exist in the simulator are aircraft database, environment database, equations of motion, X-Plane, observer panel, instructor panel, instrument display, and aircraft display in three flying modes (longitudinal, lateral, and directional). The interaction between those components is shown in Figure 3.

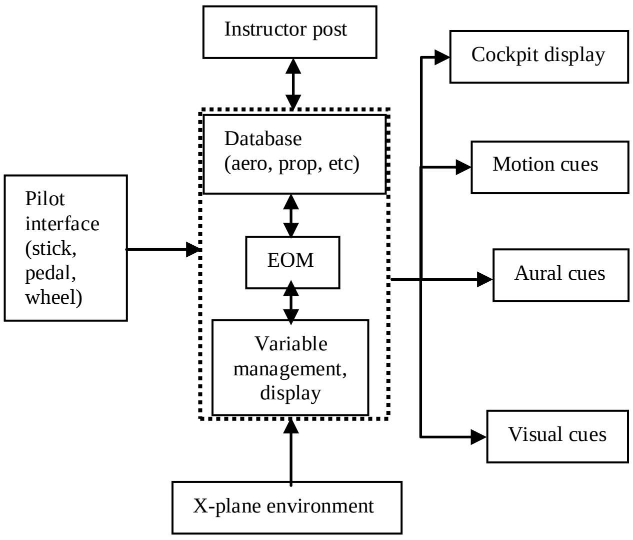

Instructor post
Cockpit display
Database (aero, prop, etc)
Motion cues
Pilot interface (stick, pedal, wheel)
EOM
Aural cues
Variable management, display
Visual cues
X-plane environment

Fig. 3: Generic aircraft simulator architecture

3. Components of Simulator Architecture

3.1. Aircraft Database

Aircraft database consists of aerodynamic parameters and configuration parameters (mass, inertia, engine position, the center of gravity location, wing area, wingspan, and mean aerodynamic chord length). Usually, we use aircraft catalog-like books in order to get configuration parameters, one of them is "Jane's All the World's Aircraft" that provides exhaustive technical detail on over 950 civil and military aircraft currently being produced or under development by more than 550 companies [7].

Most of the time, we use software called DATCOM+ in order to get aircraft aerodynamic-related data started from angle of attack and sideslip angle related parameters, dynamic characteristics, and its high lift and control devices characteristics. DATCOM+ calculate aircraft's aerodynamic characteristics based on experimental data that has been done in 1930's-1950's. Calculation result from DATCOM+ is quite accurate for conventional aircraft design especially in terms of lift and drag coefficient [10]. Another source of aerodynamic parameters is NASA's document "A Collection of Nonlinear Aircraft Simulation in Matlab" that provides needed parameters for aerodynamic force and moment modeling, but its only limited to several aircrafts needed.

The output from DATCOM+ which has a format 'filename'.out must be converted into another format that Matlab and Simulink could read using Matlab function "datcomimport('Filename.out'). After all aerodynamic parameters are defined in Matlab workspace it can be used for calculating aircraft motion in Matlab and Simulink. Here, DATCOM+ is used because it only took seconds to minute to obtained aircraft aerodynamic database. DATCOM+ also compatible with Matlab proven with matlab function as mentioned before, so database storing process could be done easily.

3.2. Equations of Motion

This is the most important part of this simulator, because we want to build generic aircraft simulation platform. We use two references for modeling aircraft motion. The first one based on "Modern Flight Dynamics" book that covers aircraft motion equation derivation until how to solve it using the simulator. Those 12 equations used are presented as follows [9]. Notation in these equations are explained in definition below:

Definition 1:

[u v w]	translational motion
[p q r]	angle rate of change
[Φ θ Ψ]	euler angle
[X_E Y_{E+} h]	position of the aircraft, I_{ij} is for inertia of the aircraft where i and/or j denote X, Y, or Z
[F_{ij} M_{ij}]	force and moment that acted on the aircraft which I denote the source of the force and moment come from and j denotes the axis
$\hat{X}$ *and* $\bar{X}$	general variable comprised in matrithe x and vector variable respectively.
Subscript "e", "a"	denote engine and aerodynamic respectively

$$\dot{u} = -qw + vr + g\sin\phi + \frac{F_{a_X}+F_{e_X}}{m} \quad (1)$$

$$\dot{v} = -ru + pw + g\cos\theta\sin\phi + \frac{F_{a_Y}+F_{e_Y}}{m} \quad (2)$$

$$\dot{w} = -pv + qu + g\cos\theta\sin\theta + \frac{F_{a_Z}+F_{e_Z}}{m} \quad (3)$$

$$\dot{p} = \frac{[qr(I_Y I_Z - {I_Z}^2 - {I_{ZX}}^2) + pqI_{ZX}(I_Z + I_X - I_Y) + LI_Z + NI_{ZX}]}{I_Z I_Z - {I_{ZX}}^2} \quad (4)$$

$$\dot{q} = \frac{[rp(I_Z - I_X) + I_{ZX}(r^2 - p^2) + M]}{I_Z I_Z} \quad (5)$$

$$\dot{r} = \frac{[qrI_{ZX}(I_Y - I_Z - I_X) + pq({I_{ZX}}^2 + {I_X}^2 - I_X I_Y) + LI_{ZX} + NI_X]}{I_Z I_X - {I_{ZX}}^2} \quad (6)$$

$$\dot{\Phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \quad (7)$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \quad (8)$$

$$\dot{\Psi} = (q\sin\phi + r\cos\phi)\sec\theta \quad (9)$$

$$\dot{X}_E = u\cos\theta\cos\Psi + v(\sin\phi\sin\theta\cos\Psi - \cos\phi\sin\Psi) + w(\cos\phi\sin\theta\cos\Psi + \sin\phi\sin\Psi) \quad (10)$$

$$\dot{Y}_E = u\cos\theta\sin\Psi + v(\sin\phi\sin\theta\sin\Psi + \cos\phi\cos\Psi) + w(\cos\phi\sin\theta\sin\Psi - \sin\phi\cos\Psi) \quad (11)$$

$$\dot{h} = u\sin\theta - v\sin\phi\cos\theta + w\cos\phi\cos\theta \quad (12)$$

Another reference that can be used comes from a thesis written by Özgür Ateşoğlu that comprised of sets of common equations of motion as shown in equation 13-16 and also more detail about engine modeling as well as its effect to aircraft motion. The next chapters discuss about uncommon fighter aircraft maneuver like Cobra Maneuver, Herbst Maneuver and how to make a controller of that motion [2], it requires a different approach of how we simulate that kind of motion,

$$\begin{matrix}[\dot{u}\ \dot{v}\ \dot{w}]^T \\ [\dot{p}\ \dot{q}\ \dot{r}]^T\end{matrix} = \hat{F} + \hat{G}\begin{bmatrix}\overline{F_1}^{(b)} \\ \overline{F_2}^{(b)}\end{bmatrix} + \bar{H}\begin{bmatrix}\overline{F_a}^{(b)} \\ \overline{M_a}^{(b)}\end{bmatrix} + \begin{bmatrix}[\bar{0}] \\ [\dot{p}_e\ \dot{q}_e\ \dot{r}_e]^T\end{bmatrix} \quad (13)$$

$$\begin{bmatrix}\dot{p}_e \\ \dot{q}_e \\ \dot{r}_e\end{bmatrix} = \hat{J}^{-1}\begin{bmatrix}0 \\ rJ_e w_e \\ -qJ_e w_e\end{bmatrix} \quad (14)$$

$$\begin{bmatrix}\dot{x}(t) \\ \dot{y}(t) \\ \dot{h}(t)\end{bmatrix} = \hat{C}^{(o,b)}(t)\begin{bmatrix}u(t) \\ v(t) \\ w(t)\end{bmatrix} - \begin{bmatrix}V_w(t)\,cos(\Psi(t)) \\ V_w(t)\,sin(\Psi(t)) \\ \dot{h}_w(t)\end{bmatrix} \quad (15)$$

$$\begin{bmatrix}\dot{\phi} \\ \dot{\theta} \\ \dot{\Psi}\end{bmatrix} = \begin{bmatrix}1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi\end{bmatrix}^{-1}\begin{bmatrix}p \\ q \\ r\end{bmatrix} \quad (16)$$

We can choose between 2 sets of equations that equations can be modeled in Matlab or Simulink by determining initial condition, and then the aerodynamics coefficients and aircraft configuration obtained from aircraft database.

3.3. X-Plane and Python Plugins

Here, X-Plane is used to visualize aircraft motion that was modeled before in Simulink or Matlab, X-Plane also has built-in basic world scenery and it still can be improved using another software. So, X-Plane played a quite important role in order to give the right visual cue to both pilot and flight test engineer even though it is only a simulator. X-Plane itself has a built-in dynamic model for each aircraft, but we want to 'inject' our own dynamic data into X-Plane, so we use a python script named PI_DataInject.py that has a function to disable built-in X-Plane aircraft dynamics that will be explained more detail later in section 4.

Fig. 4: Example X-Plane cockpit display

Python interface was made by Sandy Barbour and acted as a plugin in X-Plane to accommodate any python scripts to be executed in X-Plane. In our generic simulator platform, we made a python scripts called PI_Simmanager.py and PI_DataInject.py that made communication between X-Plane and Matlab/Simulink possible. The data sent from X-Plane are flight parameters, while from Matlab's data is user manipulation to aircraft configuration or environment setup. Overview of what Python Interface looks like in X-Plane is shown in Figures 5 and 6.

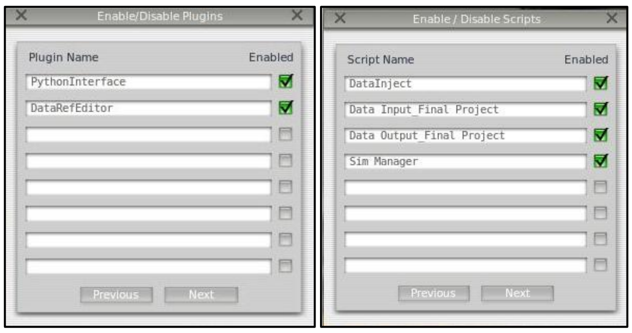


Fig. 5: Python plugin manager

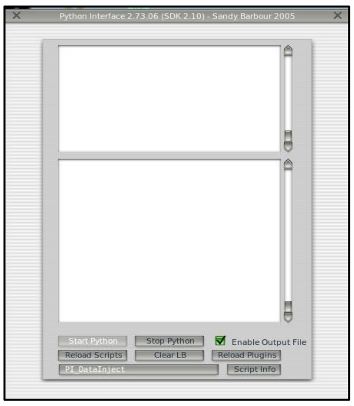


Fig. 6: Python script controller

3.4. Simulation Interface

3.4.1. Observer Panel

The aircraft observer panel is a Matlab based GUI that has a function for monitoring aircraft parameters that related to its dynamic characteristics like altitude, airspeed, groundspeed, pitch angle, flight path angle, roll angle, heading angle, track angle, deflection of aileron, rudder, and elevator. Using aircraft observer panel, instructor or flight test engineer could make an assessment of the aircraft motion qualities related to its dynamics characteristic. It can be stability, controllability, etc.

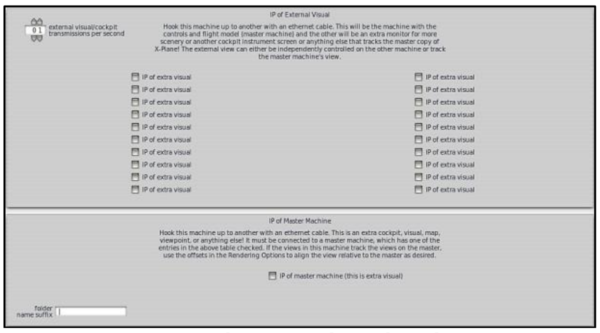
3.4.2 Instructor Panel

Instructor panel is a Matlab based GUI that has functions for monitoring and manipulating following things:
- Aircraft Loading
- Aircraft System (aircraft, instrument, engines, wings)
- Environment Setup (wind speed, wind location, wind layer, turbulence, visibility, storm)

Using instructor panel, flight test engineer or instructor can perform certainly certified flight test simulation like climb test with one engine inoperative, stall testing, out of trim testing, weight limits and center of gravity limits, etc [5].

3.4.3 Instrument Display

Aircraft instrument played a significant role in flight safety by giving important information displayed on certain instruments, like basic-T configuration that consists of crucial parameters that are an altimeter, airspeed indicator, horizontal situation indicator, and attitude director indicator [6].
Here, instrument display is separated from main cockpit view because we want to observe it thoroughly and make an assessment of aircraft attitude. To do that X-Plane already has a feature to show aircraft's instrument separately by sending it into UDP network as shown in Figure 7. We can set the IP and port that we addressed as shown in Figure 8.


IP of External Visual

01 external visual/cockpit transmissions per second

Hook this machine up to another with an ethernet cable. This will be the machine with the controls and flight model (master machine) and the other will be an extra monitor for more scenery or another cockpit instrument screen or anything else that tracks the master copy of X-Plane! The external view can either be independently controlled on the other machine or track the master machine's view.

☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual
☐ IP of extra visual ☐ IP of extra visual

IP of Master Machine

Hook this machine up to another with an ethernet cable. This is an extra cockpit, visual, map, viewpoint, or anything else! It must be connected to a master machine, which has one of the entries in the above table checked. If the views in this machine track the views on the master, use the offsets in the Rendering Options to align the view relative to the master as desired.

☐ IP of master machine (this is extra visual)

folder name suffix

Fig. 7: X-Plane external visual window

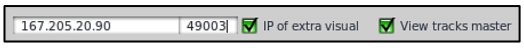

167.205.20.90 49003 ✓ IP of extra visual ✓ View tracks master

Fig. 8: IP and port determination for external view

3.4.4 Aircraft Display

To get a better view of how aircraft behaves while flying, we use three extra monitors to separate three modes of flying, they are longitudinal, lateral, and directional. The longitudinal mode is seeing the aircraft from the right wing in order to observe surging, heaving, and pitching motion. The lateral mode is seeing the aircraft from the nose or from behind to observe heaving and rolling motion. The directional mode is seeing the aircraft from above to observe surging, and yawing. Those three modes are shown in Figures 9 until 11.

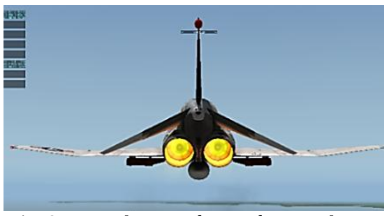


Fig. 9: Lateral view of aircraft in X-Plane

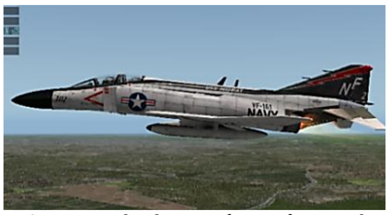


Fig. 10: Longitudinal view of aircraft in X-Plane

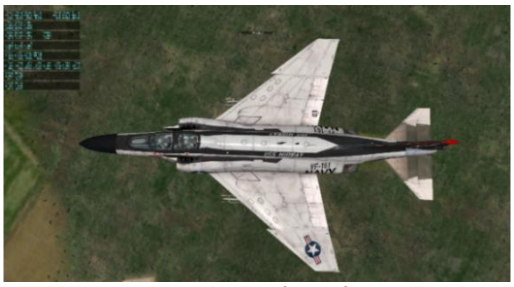


Fig. 11: Directional view of aircraft in X-Plane

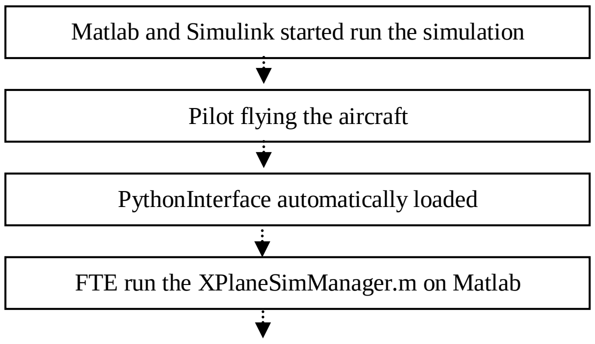
4. Current Development Stage

In the current development, we have developed few things:
- Numerical EOM model
- Aircraft database
- X-Plane and Matlab interface
- Instructor post
- Observer post

A generic simulation system that we made consists of X-Plane, Python script, Simulink, and Matlab. X-Plane as an aircraft simulator already has built-in aircraft dynamics calculation and it has a good representation of real aircraft motion. Our simulator has reached stage 6 based on development stage as shown in Figure 2.

4.1. Simulation Scheme

X-Plane modeled an aircraft motion through engineering process called "blade element theory" to calculate the force on it to get aircraft acceleration that will be integrated into velocity and position[11]. As mentioned in Figure 1, a fully developed model of simulator that currently used in our laboratory is the second one which the aircraft dynamic model is X-Plane based, while the other one is being developed. The more detail simulation platform scheme for both models in Figure 1 is presented in Figure 12 and 13 below.


Matlab and Simulink started run the simulation
Pilot flying the aircraft
PythonInterface automatically loaded
FTE run the XPlaneSimManager.m on Matlab

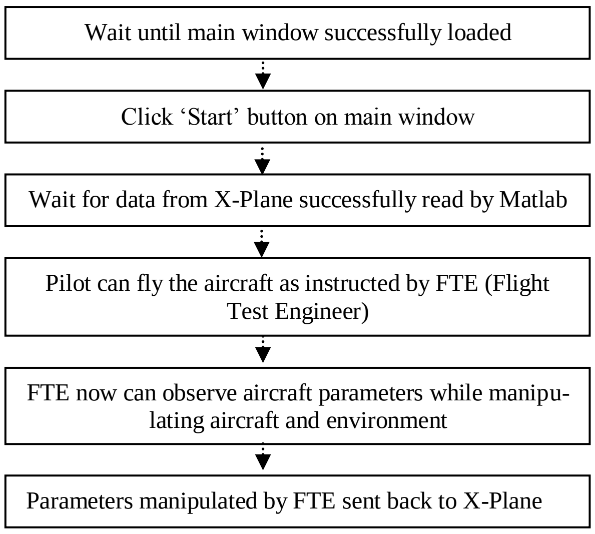

Wait until main window successfully loaded
Click 'Start' button on main window
Wait for data from X-Plane successfully read by Matlab
Pilot can fly the aircraft as instructed by FTE (Flight Test Engineer)
FTE now can observe aircraft parameters while manipulating aircraft and environment
Parameters manipulated by FTE sent back to X-Plane

Fig. 12: How this platform works

Our simulation platform should be operated with two people which act as a pilot and a flight test engineer (FTE), the objective is to keep pilot focus in operation while the flight test engineer changes the environment and observe carefully the flight parameters. The components of Figures 12 and 13 are explained in the next chapter.

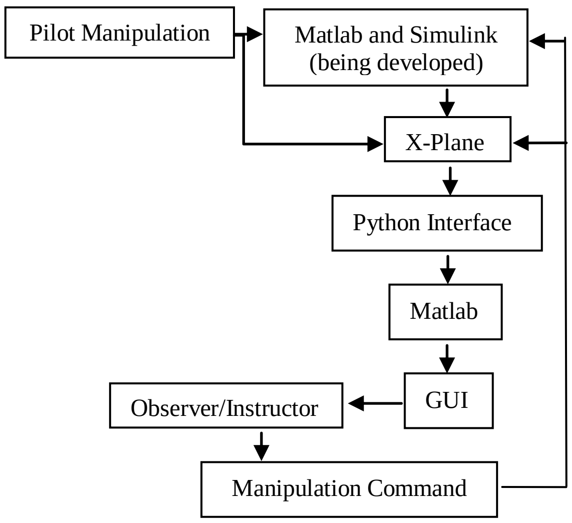


Fig. 13: How Matlab and X-Plane communicate

4.2. Numerical Components

4.2.1. Python Interface

Inside the X-Plane there are so many variables that comprise all aspects in flight classified to 18 major classes. In each class, there are also subclasses as well that we can refer to certain variables. Those variables are called DataRefs. For example, we want to know how many engines used by the airplane at that moment, so we have to refer to sim/aircraft/engine/acf_num_engines. The first section "sim" is always like that, then the second section "aircraft" is the major classes, the third one "engine" is the subclass of "aircraft" class and then finally the last one "acf_num_engines" is the variable that we want to refer to, however some variables do not have subclasses.

In addition, as mentioned in chapter 3, our objectives are to be able to model any kinds of aircraft and using our own aircraft database. So in order to do that, we have to disable X-Plane built-in aircraft dynamics using python script named PI_DataInject.py that has following pseudo-code architecture in Figure 14.

```
/calling libraries/
...
/calling datarefs → planepath, simulation time, and control surface/
...
Loop:       /freeze → planepath, simulation time, and control surface/
            ...
            /activate port for UDP communication data/
            ...
            /sending or receiving UDP data to assign port/
```

Fig. 14: Pseudo code for injecting aircraft motion from Simulink

After communication between Matlab, Simulink, and X-Plane are established in term of aircraft motion and flightpath now we proceed to execute observer and instructor platform, first we have to know what variables that we want to observe and manipulated, after that, we use a python script named PI_SimManager.py to do data sending-receiving between X-Plane and Matlab using following pseudo code in Figure 15.

```
/Calling libraries/
...
/Defining Port for Communication/
...
Loop:       /Defining DataRefs that will be sent/received to certain port/
            ...
            /Sending Datarefs to certain port/
            ...
            /Receiving UDP data from Matlab/
            ...
            /Assigning received data to Datarefs/
```

Fig. 15: Pseudo Python Script used for X-Plane and Matlab Communication

4.2.2 Matlab

The Data that had been sent over UDP network using PI_SimManager.py are received using another script written using Matlab language named XPlaneSimManager.m, the script itself contain following pseudo code as shown in Figure 16.

```
/Declaring ports for UDP communication/
...
/Create GUI → main window/
...
/Create GUI → aircraft and environment tabs/
...
Loop:       /Receiving data from UDP network/
            ...
            /Create a plot in Aircraft Observer tab each delta time/
            ...
            /Sending data UDP network to X-Plane if user give inputs/
            ...
            /Update aircraft observer chart/
```

Fig. 16: Pseudo Matlab Script for Creating GUI and User Manipulation

Using Matlab GUI that the pseudo code has been explained in the previous chapter, figures below presented how those GUI look like and its function as well.

Figure 17 shows aircraft observer window in action. This window observes flight parameters like altitude (ft), airspeed (knots), roll angle (degree), pitch angle (degree), gamma angle (degree), sideslip angle (degree), track angle (degree), and aileron, rudder, elevator deflection (degree).

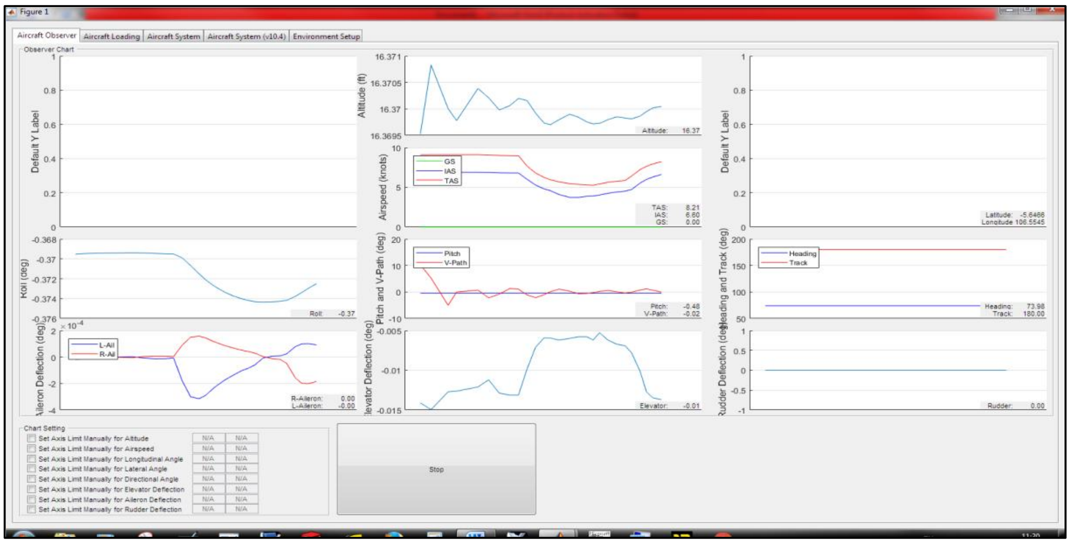
Figure 18 depicts instructor window which instructor could change CG shift, payload weight, fuel weight distribution, and then send its final configuration to X-Plane by clicking "set AC loading" button.

Figure 19 shows aircraft system manipulator window. It has a function to manipulate aircraft systems functionality. The aircraft system(s) can be set normally operative up to total failure. For example, setting the engine to inoperative until a certain minute, jamming the control surface, etc.

Figure 20 shows aircraft system manipulator window as well, but it is for another version of X-Plane. It has the same function of aircraft manipulator window in Figure 19.

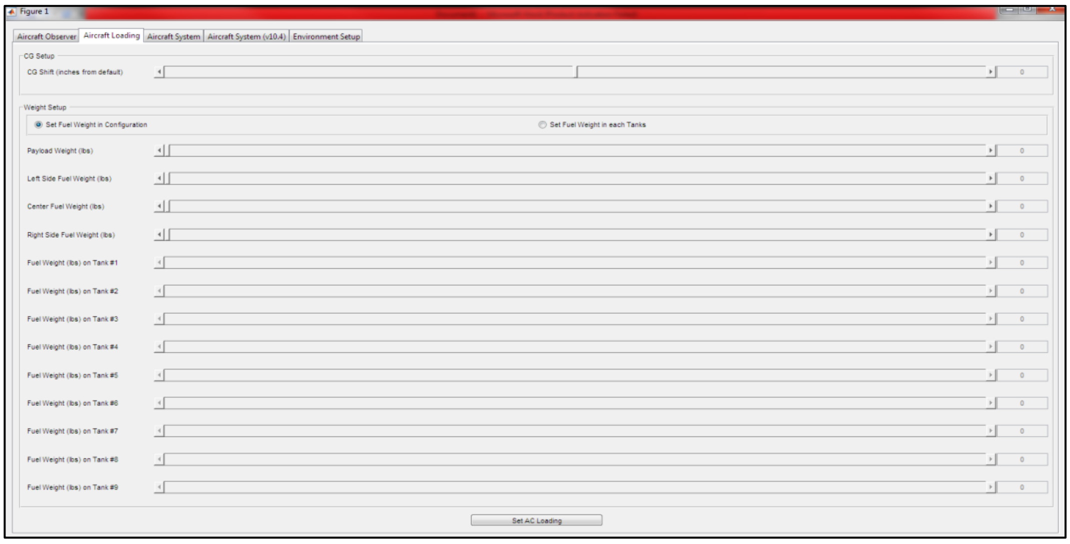
Figure 21 shows environment manipulator window that one can manipulate visibility, precipitation, storms, turbulence, wind speed, wind direction, wind gust and its location as well.

4.2.3 GUI for Observer and Instructor



Set Axis Limit Manually for Altitude N/A N/A
Set Axis Limit Manually for Airspeed N/A N/A
Set Axis Limit Manually for Longitudinal Angle N/A N/A
Set Axis Limit Manually for Lateral Angle N/A N/A
Set Axis Limit Manually for Directional Angle N/A N/A
Set Axis Limit Manually for Elevator Deflection N/A N/A
Set Axis Limit Manually for Aileron Deflection N/A N/A
Set Axis Limit Manually for Rudder Deflection N/A N/A

Fig. 17: Aircraft observer window in plotting received data



CG Setup

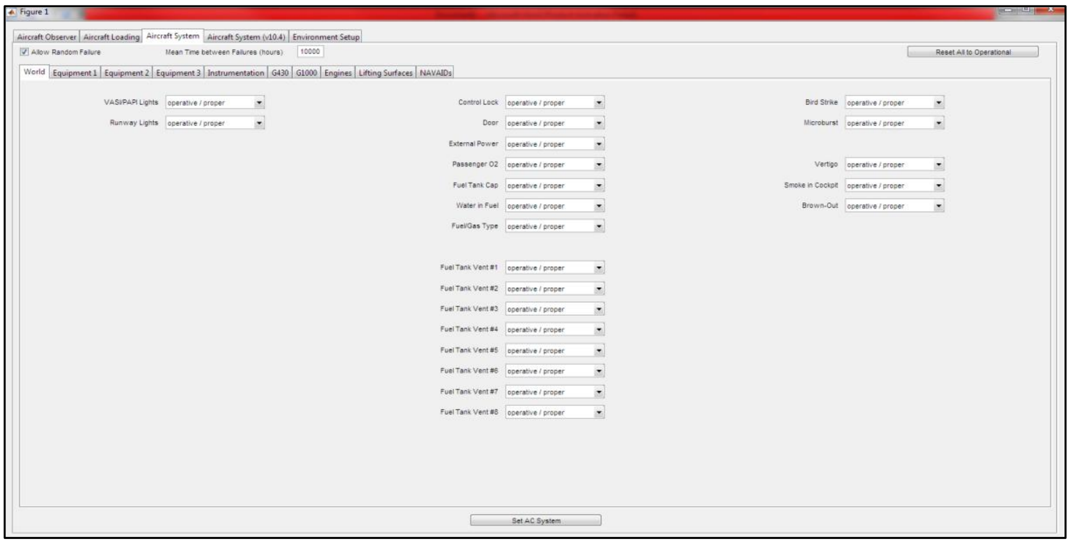
CG Shift (inches from default)	0

Weight Setup

(● Set Fuel Weight in Configuration) Set Fuel Weight in each Tanks

Payload Weight (lbs)	0
Left Side Fuel Weight (lbs)	0
Center Fuel Weight (lbs)	0
Right Side Fuel Weight (lbs)	0
Fuel Weight (lbs) on Tank #1	0
Fuel Weight (lbs) on Tank #2	0
Fuel Weight (lbs) on Tank #3	0
Fuel Weight (lbs) on Tank #4	0
Fuel Weight (lbs) on Tank #5	0
Fuel Weight (lbs) on Tank #6	0
Fuel Weight (lbs) on Tank #7	0
Fuel Weight (lbs) on Tank #8	0
Fuel Weight (lbs) on Tank #9	0

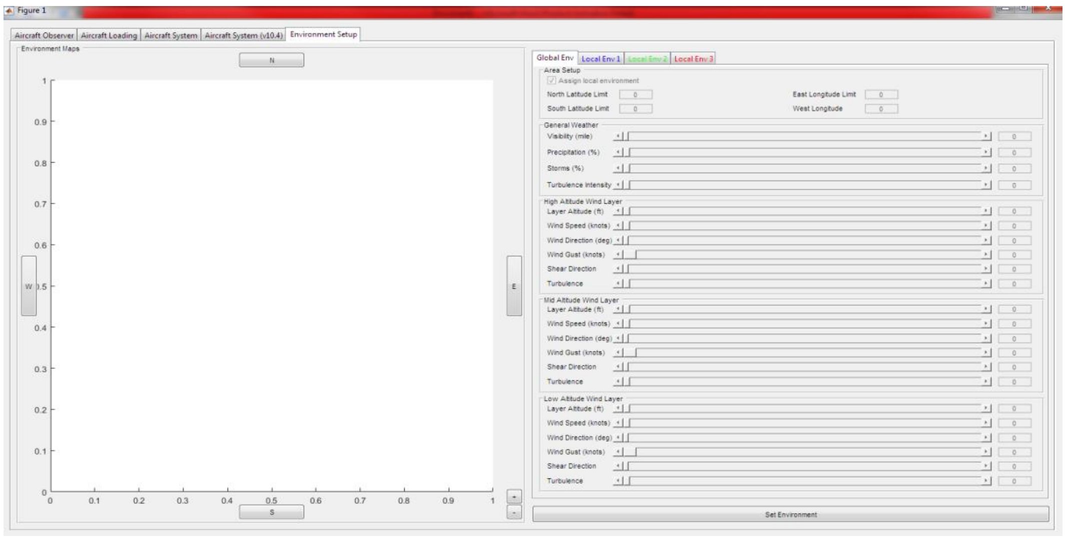
Fig. 18: Aircraft loading window



☑ Allow Random Failure Mean Time between Failures (hours) 10000

VASI/PAPI Lights	operative / proper
Runway Lights	operative / proper
Control Lock	operative / proper
Door	operative / proper
External Power	operative / proper
Passenger O2	operative / proper
Fuel Tank Cap	operative / proper
Water in Fuel	operative / proper
Fuel/Gas Type	operative / proper
Fuel Tank Vent #1	operative / proper
Fuel Tank Vent #2	operative / proper
Fuel Tank Vent #3	operative / proper
Fuel Tank Vent #4	operative / proper
Fuel Tank Vent #5	operative / proper
Fuel Tank Vent #6	operative / proper
Fuel Tank Vent #7	operative / proper
Fuel Tank Vent #8	operative / proper
Bird Strike	operative / proper
Microburst	operative / proper
Vertigo	operative / proper
Smoke in Cockpit	operative / proper
Brown-Out	operative / proper

Fig. 19: Aircraft failure scenario window



Area Setup
☑ Assign local environment
North Latitude Limit 0 East Longitude Limit 0
South Latitude Limit 0 West Longitude 0

General Weather
Visibility (mile) 0
Precipitation (%) 0
Storms (%) 0
Turbulence Intensity 0

High Altitude Wind Layer
Layer Altitude (ft) 0
Wind Speed (knots) 0
Wind Direction (deg) 0
Wind Gust (knots) 0
Shear Direction 0
Turbulence 0

Mid Altitude Wind Layer
Layer Altitude (ft) 0
Wind Speed (knots) 0
Wind Direction (deg) 0
Wind Gust (knots) 0
Shear Direction 0
Turbulence 0

Low Altitude Wind Layer
Layer Altitude (ft) 0
Wind Speed (knots) 0
Wind Direction (deg) 0
Wind Gust (knots) 0
Shear Direction 0
Turbulence 0

Set Environment

Fig. 20: Environment control window

4.2.4. Simulink Model

Another scheme of our simulator is using a dynamic model built in Simulink which is later visualized in X-Plane using python plugin as mentioned in Figure 14. After built-in aircraft dynamics in the X-Plane is disabled, we can inject our aircraft dynamics modeled in Matlab and Simulink. First, we convert DATCOM+ output file into a variable stored in the workspace as explained in chapter 3. After that, we build equations of motion blocks in Simulink that comprises aircraft database (dimension and aerodynamics) as shown in Figure 21.

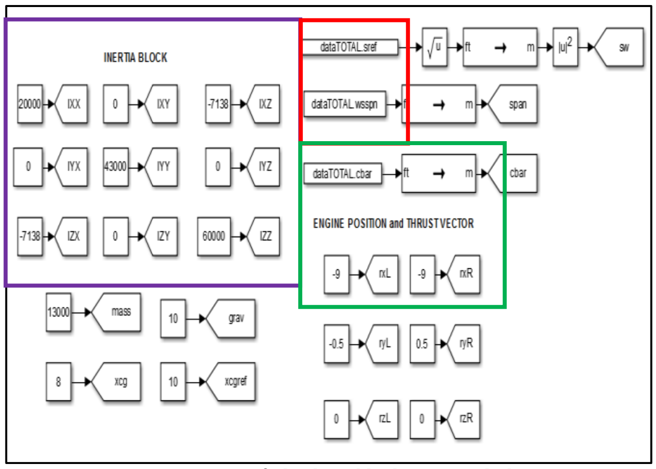


Fig. 21: Aircraft database blocks in Simulink

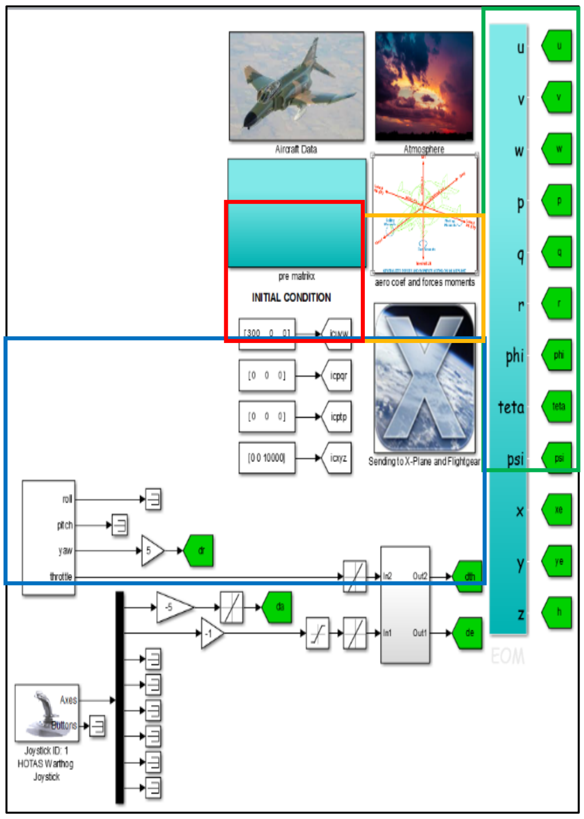


Fig. 22: Overview of aircraft modeling using Simulink block

The red square in Figure 21 is an example of using aircraft database from Matlab workspace, the purple square is an area of aircraft inertia database, and the green square area is engine location database. After all aircraft database is well established in Simulink, we can add atmospheric condition using default Simulink block, and then we can model its dynamic motion based on initial condition and user input utilizing joystick and pedals.

Overview of Simulink blocks that we used is shown in Figure 22, the green square consists of twelve equations of motion, the blue square is consist of block that receives input from joystick and pedal and converts it into numeric values to manipulate control surfaces deflection. Orange block has a function to send aircraft's Euler angle data in degree and its position in latitude-longitude. Lastly red square area is block to determine initial condition for twelve sets equation of motion in (1) – (12). After Simulink blocks are well established, simulation can be run for an infinite time as long as needed with X-Plane is ready to use before it with condition an aircraft already selected and located in a runway.

Simulation scheme in chapter 4.2.4 is successfully executed in term of Simulink to X-Plane, the vice versa communication between X-Plane and Matlab using own dynamic model from Simulink is being developed as well as its observer and instructor station because current observer and instructor station platform communicate directly between Matlab and X-Plane only.

4.3. Simulation Hardware

Figure 23 shows aircraft observer and instructor window in operation while X-Plane is in operating mode as well. Here observer and instructor post is located directly below X-Plane window for the sake of clarity. Usually we used the large monitor in separate desktop/room to place aircraft observer and instructor post. We use few types of equipment to make this simulator resemble actual cockpit environment which will show in figures below.

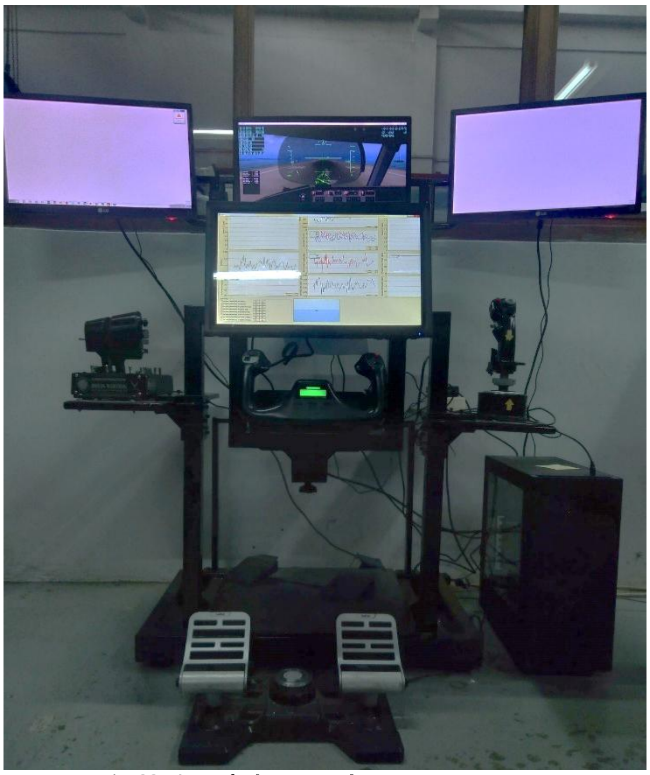


Fig. 23: Aircraft observer and instructor in operation

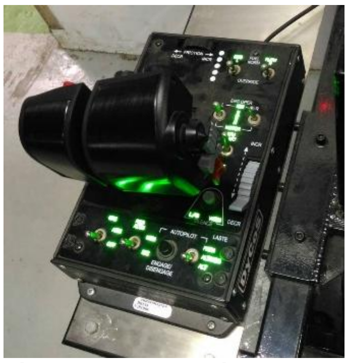


Fig. 24: Thrustmaster Hotas Warthog Engine Controller

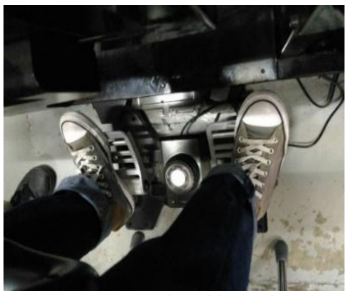
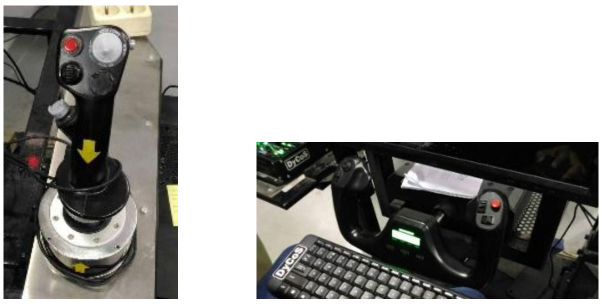


Fig. 25: Saitek rudder pedal


(a) (b)
Fig. 26: (a) Thrustmaster Hotas Warthog Joystick (b) Saitek yoke stick

4.4. Simulation Example

Below we presented few examples of simulation in three flight cases, cruising, climbing, and turning with synchronized figures from X-Plane and Matlab GUI.

4.4.1. Cruising

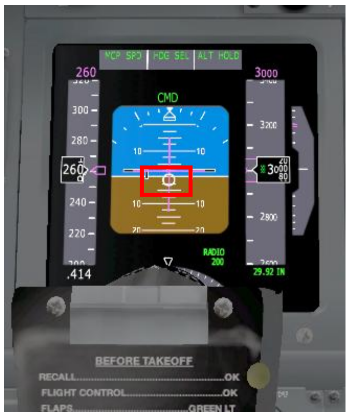


Fig 27: An aircraft cruising in X-Plane indicating with flightpath indicator (white circle in red square) lies on artificial horizon indicator. And the Vertical Speed Indicator is zero.

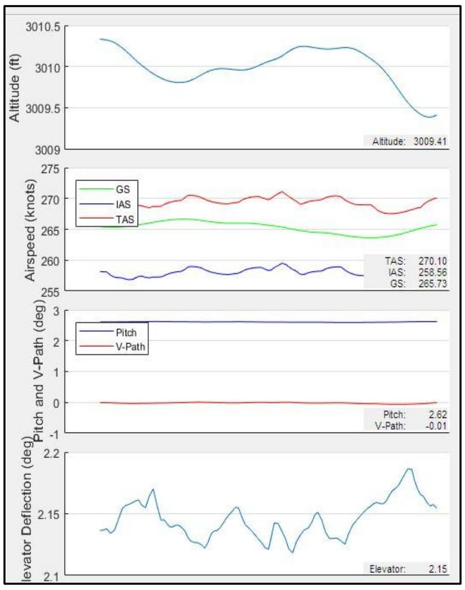


Fig 28: Observer window read aircraft data while cruising shown in flightpath angle is 0 degree.

4.4.2. Climbing

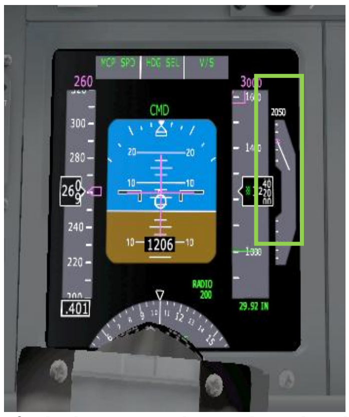


Fig 29: An aircraft climbing in X-Plane indicated with Vertical Speed Indicator shows positive values (green square).

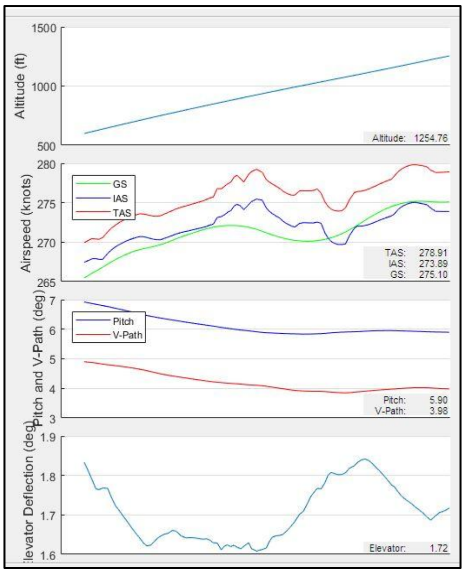


Fig 30: Aircraft observer shows the aircraft make a climb maneuver indicated with positive flightpath angle and increasing altitude as well.

4.4.3. Turning

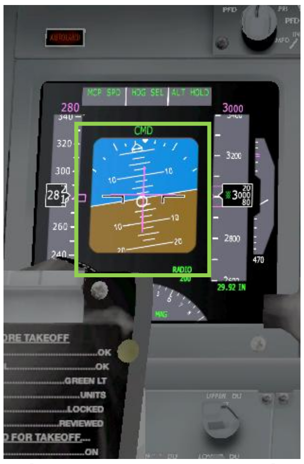


Fig 31: An aircraft makes a turn maneuver in X-Plane indicated with roll attitude indicator tilt to right (green square).

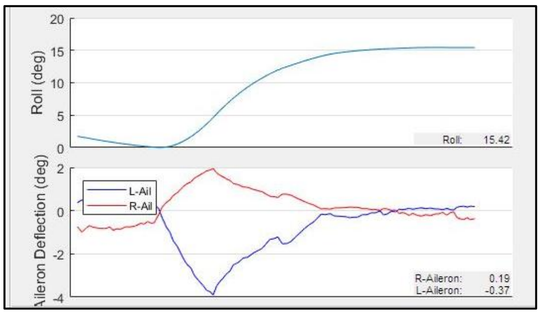


Fig 32: Aircraft observer shows the aircraft make a turn by banking about 15 degrees.

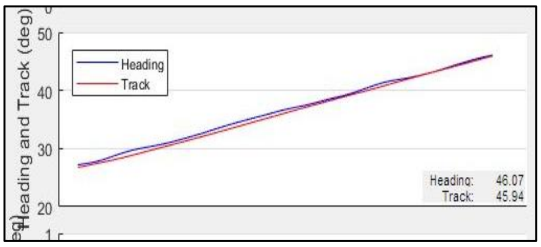


Fig 33: Aircraft observer shows the aircraft change its heading by banking about 15 degrees as shown in Figure 34

5. Conclusions

Currently aircraft simulator that is being developed by ITB (Flight Physics Research Group) consists of aerodynamic and engine model, instructor station, and observer station, using Matlab based on X-Plane's aircraft dynamics, and equation of motion modeled in Simulink. The simulation platform type one, which consists of X-Plane as a main simulator, Python interface, and Matlab as an observer and instructor station, works perfectly as explained in section 4 together with a few simulation examples. Meanwhile simulation platform type two, which utilizes Simulink as a main software to model the aircraft dynamics is being developed. So far we have developed equation of motion in Simulink for any kind of aircraft, Python script that turn off X-Plane based aircraft dynamics, and Simulink blocks, that send Euler angle and aircraft position to X-Plane for visualization. As the future works, the equation of motion modeled in Simulink can be modified to accommodate aircraft motion if the engines or control devices have failed. For example, if the right elevator does not work, there will be rolling moment as well, so the aileron must compensate for it. The observer and instructor station for Simulink based model have not been made. Finally, to achieve the main goal of Flight Physics Research Group, integration between flight numerical simulation and miniature of motion platform should be done.

Acknowledgement

This work is a part of research funded by the Ministry of Research and Higher Education (Kemenristekdikti) Republic of Indonesia.

References

[1] Allerton, D. Principles of Flight Simulation. (2009). Chichester: John Wiley & Sons, Ltd. 1-8
[2] Ateşoğlu, Ö. High Angle of Attack Maneuvering and Stabilization Control of Aircraft. (2007). Turkey: Middle East Technical University. 62-85
[3] Barbour, S. (2013, 07 18). Pyton Interface. Retrieved from Sandy Barbour's XPlugin SDK Website: http://www.xpluginsdk.org/python_interface.htm
[4] Contributors, W. (2018, April 5). X-Plane (simulator). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/X-Plane_(simulator)
[5] Federal Aviation Administration. (12). Advisory Circular, Flight Test Guide For Certification of Transport Category Airplanes. Washington: U.S Department of Transportation.
[6] Ian Moir, A. S. Civil Avionics System. (2013). Chichester: Wiley. 453-455
[7] Jackson, P. Jane's All the World Aircraft. (2004). -: Jane's Publishing.
[8] MIL-F-8785C. Military Spesification.
[9] Schmidt, D. K. Modern Flight Dynamics. (2012). New York: McGrawHill. 445-451
[10] Thomas S. Richardson, C. B. Analysis of the Boeing 747-100 using CEASIOM. (2011). Progress in Aerospace Sciences, 666-672.
[11] X-Plane. How X-Plane Works. (2018). Retrieved from X-Plane 11: http://www.x-plane.com/desktop/how-x-plane-works/