Modeling and Autonomous Flight Simulation of a Small Unmanned Aerial Vehicle

T. A. M. Abdunabi*, M. El-Gelani**, N. M. Nasr***

Abstract: This paper describes the use of FlightGear, an open-source flight simulator, and JSBSim, an open source flight dynamics model, to model and simulate a small autonomous Unmanned Aerial Vehicle (UAV). A small commercial electric engine Cessna-182 radio controlled (RC) aircraft was chosen to represent the UAV. The first step was to create the required JSBSim aircraft configuration files by using the Aeromatic v0.8, a free web application to create aircraft configuration files for use with the JSBSim. The next step was to make educated guesses to refine important sections in the created configuration files with the assistance of available data of similar UAV. In order to perform a visual simulation, a 3D model for the Cessna-182 (RC) was created using AC3D, a commercial 3D modeling software tool. To fly the modeled UAV autonomously a tuning process was made for the built-in generic PID (proportional, integral, and derivative) autopilot of FlightGear, which has the ability to hold aircraft velocity, vertical aircraft speed, altitude, pitch angle, angle of attack, bank angle, and true heading. Finally, a flight path, which contains a number of waypoints chosen over a selected area using Google Earth map, was constructed. In order to use the chosen waypoints with FlightGear navigation system, a unique ID was assigned to each waypoint, and the FlightGear database was altered to include the new waypoints with their IDs. The outcome of the paper was a complete JSBSim flight dynamic model for the Cessna-182 (RC), with 3D model for visual simulation and an effective autopilot. A good autonomous flight simulation was performed. This paper concluded that modeling and simulating a UAV accurately is not an easy task, due to the need to calculate many parameters either by physical measurements, experiments, or estimation from available data of similar UAV, or by software tools.

Keywords: UAV Modeling, UAV Simulation, JSBSim, FlightGear

1. Introduction

Small Unmanned Aerial Vehicles (UAVs) are increasingly used by researchers, hobbyists, civilian organizations, and the military for different purposes, due to the lack of risk to a pilot and their low cost. In some cases, these UAVs are flown autonomously by on-board autopilots, with highly integrated systems and complex control laws. The use of simulation environments to test the UAVs design and control systems is an important phase in their development cycle, since it reduces the time and the risks associated with the real flight [1].
The aims of this paper are to investigate how to create a Flight Dynamic Model for a given small UAV, and how to visualize that model in a Flight Simulator with the ability to perform autonomous flight simulation.

The approach followed in this paper should be clear and relatively easy in order to allow the students and researchers to follow it in their research. A commercial small Cessna 182 radio controlled aircraft shown in Fig. 1 was considered to be the starting point of the project.

![Cessna 182 RC](image)

**Fig. 1 Cessna 182 RC**

To achieve the aims of the paper the following related items to Cessna 182 RC must be delivered:

- A complete Flight Dynamic Model
- A 3D graphical model
- An effective autopilot
- A way to identify flight route
- Autonomous flight simulation

2. **A Review of Flight Simulators**

A flight simulator is a system that attempts to imitate the experience of flying an aircraft as practically as possible. The types of flight simulators vary from personal computer video games to full size cockpit replicas controlled by sophisticated computer systems. Flight simulators are widely used by academic researchers, the aviation industry, and air forces for pilot training and aircraft development.

This section reviews some of the most relevant flight simulators available for use, explaining the advantages and disadvantages of using each, and justifies the chosen flight simulators. Two flight dynamic model (FDM) simulators, AeroSim blockSet, and JSBSim, and two visualization flight simulators, FlightGear and Microsoft Flight Simulator, have been reviewed.

2.1. **AeroSim BlockSet**

The AeroSim blockSet is a Matlab/Simulink block library designed by flight control engineers, and provides elements for fast development of nonlinear 6-degree-of-freedom (6-DOF) aircraft dynamic models. Academic and non-commercial users can download the AeroSim blockSet freely for use. The Aerosim blockSet features are [2]:

2/10
Non-linear 6-DOF aircraft dynamics are implemented in various reference frames.
Linear aerodynamics, piston-engine propulsion, and time-varying inertia models are provided.
The model can read the aerodynamic, propulsion, and inertia data from user-defined sources.
Standard atmosphere model, wind shear, background wind, and turbulence models are available.
Detailed earth models are provided.
Transformations to and from multiple reference frames are provided.
Conversions between engineering units are provided.
XML aircraft configuration files of JSBSim flight dynamic model can be used.
Pilot interface blocks for joystick input and for visual output in FlightGear Flight Simulator or Microsoft Flight Simulator are provided.
Pre-built aircraft models, which can be reused by changing the parameters file, are provided.
By using the Real-Time Workshop, which is one of the Matlab tools, C language code can be generated automatically from the model.

The main disadvantage of AeroSim blockSet is that it does not model aircraft with an electric propulsion system.

2.2. JSBSim Flight Dynamic Model version 2.0
JSBSim is an open source flight dynamic model (FDM) written in the C++ programming language. It is the default flight dynamic model for the FlightGear flight simulator.

JSBSim Features:
- Compiles and runs under many operating systems.
- Can be run as a stand-alone program, or it can be run as an integrated part of the flight simulator, which provides visual output.
- The code is generic, and particular aircraft flight control systems, propulsion, aerodynamics, landing gears, and autopilot are defined in XML format configuration files.
- Can be used to model many propulsion systems including an electric propulsion system.
- Models the rotational earth effects on the equation of motion.
- Many data output formats such as to screen, socket, file.

One of the strongest advantages of the JSBSim is that it is an open source project, which gives a very flexible environment for academic researchers and volunteers around the world to use it as a tool in their research or to develop it itself [3].

2.3. FlightGear Flight Simulator version 0.9.10
FlightGear is an open-source flight simulator written in the C++ programming language, available on the World Web Wide. The FlightGear features [4]:
- High Degree of Freedom: FlightGear is open source, its source code can be downloaded and altered to meet a specific research purpose.
• Flight Dynamics Models: FlightGear has three primary flight dynamics models; JSBSim, YASim and UIUC- the researchers can use any one according to their requirements. In addition, it is possible to add a new dynamic model or interface with an external model.
• Extensive and accurate world scenery database.
• Accurate and Detailed Sky Model: FlightGear provides accurate sun, moon stars and planets, positions for a certain time and date.
• Flexible and Open Aircraft Modeling System: FlightGear can model a wide variety of aircrafts. It has very good instrument animation and infrastructure to build 3d cockpits. Moreover, it models real world instrument behavior, and system failures.
• Moderate Hardware Requirements: FlightGear can be run on a personal computer.
• Internal Properties Exposed: FlightGear allows the user to access the internal properties too and monitor any of its internal state variables. By editing configuration files it is possible to create sound effects, model animations, instrument animations and network protocols for approximately any situation.
• Networking Options: By setting its networking options, FlightGear can communicate with external flight dynamics models, GPS receivers, other instances of FlightGear, and other software.

2.4. Microsoft Flight Simulator 2004
Microsoft flight simulator (MSFS) 2004 is a popular commercial program; it is available in most computer games stores.

MSFS 2004 Features:
• Contains many aircraft models.
• Provides a tool to generate scenery automatically.
• Provides detailed visual effects.
• Provides detailed flight information using flight analysis maps and graphs.
• Provides 3D interactive cockpits.
• Provides moving maps and GPS positioning system.
• Provides multiplayer capability.

The main disadvantage of Microsoft Flight Simulator is that it is only for the Windows operating system [5].

2.5. The Chosen Flight Simulators
The JSBSim was chosen to be the flight dynamic model due to the following reasons:
• The UAV to be modeled has an electric propulsion system, which the AeroSim blockSet does not model.
• The JSBSim is the default flight dynamic model for the FlightGear flight simulator, which was chosen to visualize the output, which is an important factor in terms of computer performance.

The FlightGear was chosen to be the flight simulator instead of Microsoft flight simulator due to the following reasons:
• FlightGear is an open source, can be downloaded from the internet and used freely.
• The FlightGear source code can be altered according to the requirements of the project.
• FlightGear uses the JSBSim flight dynamic model as a default model. JSBSim is installed automatically with FlightGear.

3. Description and Construction of the Required Files to Model the UAV and Perform the Simulation

Many issues were involved in modeling the UAV. One of the big challenges was calculating its aerodynamics coefficients. Usually these coefficients are calculated by experiments, by using computer software, or from available data of similar aircraft.

Many approaches were investigated such as using DATCOM+, which is a computer program to estimate the design aerodynamics coefficients of an aircraft; however, this approach was ignored due to the complexity of the program and the need for many refinements to the program output.

The next approach was to use Aeromatic version 0.8, a web application to create aircraft configuration files for use with the JSBSim Flight Dynamics Model. According to the Aeromatic web site [6], the output aircraft configuration files format is JSBSim version 2.0. However, this is not the case at the time of constructing the Cessna-182 (RC) configuration files. The output format was JSBSim version 1.65, consequently, the UAV configuration files were rewritten in version 2.0 format.

Although, the Aeromatic gives a fast and effective way to construct the UAV configuration files, some important refinements are required. Due to that, another assistance approach had to be found.

The use of available data of similar UAV was the missing part that was required to get a good JSBSim model. Unfortunately, there are no many UAV JSBSim models available; however, the Rascal 110 R/C UAV, which is quite similar to the Cessna-182 (RC) in shape, and bigger in size, was very helpful in refining the Cessna-182 (RC) configuration files.

3.1 FlightGear and JSBSim Required Files

The FlightGear flight simulator needs essential files arranged in folders in order to perform successful simulation. This section represents the top level of all the required files.

3.1.1 UAV Main Folder

The name of this folder is the name of the modeled UAV (RC-Cessna-182). It contains the UAV main (c182.xml) and Set (c182-set.xml) JSBSim configuration files, Engines folder, Models folder, and systems folder. The main UAV folder should be the directory: $FG_ROOT\FlightGear\data\Aircraft$

3.1.2 Engines Folder

The Engines folder contains the engine (Cessna_RC_Engine.xml), and the thruster (Cessna_RC Propeller.xml) JSBSim configuration files.
3.1.3 Systems Folder
The systems folder contains the autopilot (Cessna182-auropilot.xml) and the electric system (electrical.xml) JSBSim files.

3.1.4 Models Folder
The Models folder contains the JSBSim 3D graphical model configuration file (c182-dpm.xml), the 3D graphical model (c182.ac), and textures (c182-01.rgb and c182-02.rgb) used by the 3D model.

3.2 FlightGear Autopilot and Flight Path
To perform autonomous flight simulation autopilot configuration file must be added and a waypoint must be calculated to construct the UAV flight path.

3.2.1 FlightGear Autopilot
FlightGear uses a PID (Proportional, Integral, and Derivative) algorithm designed by Roy Ovesen. FlightGear implements the algorithm in a flexible way, which makes it reusable with similar aircrafts. Any number of PID controllers can be defined in the autopilot configuration file. Each controller can be assigned a process value, reference point, any number of output values, and other tuning constants. Moreover, cascading controllers can be implemented by specifying multiple PID controllers, in which the output of the current stage is used as the input to the next stage [7].

According to [7], the best start to constructing the autopilot configuration file for the modeled aircraft is by copying the autopilot configuration file from an existing, similar aircraft, and tuning the autopilot parameters to adapt to the modeled aircraft. The most basic method of tuning autopilot parameters is the trial and error method. In this method the proportional gain, integral time, and derivative time are adjusted until the performance is acceptable.

3.2.2 Constructing the UAV Flight Path (Route)
FlightGear uses for navigation fixed waypoints such as airports and navigation aids such as radio stations. The fixed points are determined by latitude and longitude. FlightGear uses a database created by Robin A. Peel, in which a unique ID and its latitude and longitude coordinates identify each waypoint. When a waypoint is entered in the aircraft route during the simulation time, FlightGear checks this database to see if it is a valid fixed point or not. This data is stored in the compressed file called fix.dat, which can be found in the directory FG_ROOT\FlightGear\data\Nav aids.

Because FlightGear is used to replicate the real navigation system around the world, it was clear that none of the waypoints stored in the FlightGear fix database were suitable for the Cessna 182 RC UAV. Consequently, a way of calculating the UAV flight route had to be found.

Since the UAV is a small aircraft and has a limited range, it was logical that the flight route should be short. The free software map Google Earth provides a very helpful way to calculate the waypoints of the flight route. In order to deploy the calculated waypoints with FlightGear, the fix.dat database had to be altered by adding those waypoints with their IDs.
4. Running and Testing the Simulation
To run the modeled Cessna 182 RC UAV simulation FlightGear flight simulator version 0.9.10 should be installed and configured properly. The modeled UAV main folder (Rc-Cessna-182) with its contents should be placed in the directory:

FG_ROOT\FlightGear\data\Aircraft\Rc-Cessna-182

After starting the FlightGear flight simulator, selecting the Cessna 182 (RC) UAV, selecting the take off location, setting the simulation’s parameters, setting the flight dynamic model, and setting latitude and longitude coordinates of the takeoff point, the simulation can be run from the main settings window by clicking on the Run button. Figure 2 shows the Cessna 182 RC on the ground before takeoff.

![Fig. 2  Cessna 182 RC before takeoff](image)

The autopilot settings can be specified from the autopilot settings window shown in Fig. 3.

![Fig. 3  Autopilot settings window](image)

Waypoints can be specified before or during the flight from the add waypoint window, as shown in Fig. 4. It is also possible to remove any waypoint during the flight.
After setting the autopilot and adding the waypoints to the flight path, an autonomous flight simulation was ready to test. The autonomous simulation was performed with very satisfactory performance. For instance, autopilot can control the UAV to ascend to the specified altitude and hold it, as shown in Fig. 5 and Fig. 6 by activating the altitude hold in the autopilot settings window.
It is also possible to hold a constant speed, zero bank angle, pitch angle, angle of attack. To force the UAV to follow the selected waypoints, the true heading should be activated in the autopilot settings window. Figure 7 shows the Cessna 182 following the waypoints listed in the top left of the screen.

![Fig. 7 Cessna 182 RC in True heading flight](image.png)

5. Conclusion

The combination of FlightGear Flight Simulator and JSBSim Flight Dynamic model, which are open source projects written in the C++ programming language, provided a solid base for building the simulation environment.

To model the Cessna 182 RC UAV (or any aircraft) in JSBSim Flight Dynamic Model and simulate it with FlightGear flight simulator, essential configuration files must be constructed. These files include main Cessna 182 configuration, engine configuration, propeller configuration, electric system configuration, and 3D model configuration, for an autonomous flight autopilot configuration file is required. All these files are tied together in the top level configuration file.

The real challenge was to construct an accurate model for the Cessna 182 RC due to the need to estimate its parameters as accurately as possible. Some of these parameters were measured physically from the Cessna 182 RC UAV, and others were generated by the free web application Aeromatic v8.0. However, not all parameters could be generated by Aeromatic, therefore, the similar UAV Rascal 110 RC was used to estimate the missing parameters, and to refine some parameters. The constructed configuration files format should be the format of JSBSim version 2.0 in order to work with the current version of FlightGear and JSBSim.

In order to visualize the JSBSim model output of the Cessna 182 RC UAV with FlightGear, a 3D graphical model was constructed by using the commercial 3D modeling software AC3D. The 3D graphical model of the real Cessna 182, which is available to download from the FlightGear web page, was the starting point for constructing a 3D graphical model of the
Cessna 182 RC UAV. The model was re-sized to fit the UAV dimensions, and a quite tedious process was performed to allocate the 3D model within FlightGear.

The autopilot configuration file was constructed with the help of the Rascal 110 RC autopilot configuration file. The tuning process involved changing the autopilot constants until an acceptable performance was achieved.

The flight path needed to be followed by the autopilot was constructed from waypoints calculated over a selected area by using the free software map, Google Earth. Then the calculated waypoints were deployed in the FlightGear navigation system.

By combining all the files and arranging them in specific folders within FlightGear flight simulator, the simulation was performed perfectly, and the modeled Cessna 182 RC UAV was flown normally by an operator, and autonomously by its autopilot. The autopilot has the capability to perform many tasks such as altitude, velocity hold, bank angle, pitch angle, and true heading hold.

6. Future Work
This paper opens the doors to future work in performing Hardware-In-The-Loop (HITL) simulation, in which the physical system, for instance onboard autopilot, to be tested is fooled into thinking that it is working with real inputs and outputs. This can be done by connecting the autopilot hardware to FlightGear flight simulator and configuring it to control the Cessna 182 RC UAV instead of FlightGear autopilot.

7. References