3D Simulation of Flight Control System for Quad Tilt Rotor UAV Based on Flightgear

Zhigang Wang* and Jianbo Li*
National Key Laboratory of Rotorcraft Aeromechanics, Nanjing University of Aeronautics and Astronautics, Nanjing, China

*Corresponding author e-mail: ljb101@nuaa.edu.cn, *wangzhigang@nuaa.edu.cn

Abstract. The quad tilt rotor (QTR) has complex dynamics characteristics, especially in conversion mode. It is difficult to model the QTR dynamics and the environmental factors have a great influence on it. To solve the problem of control in conversion mode of QTR, this paper carries out the design of the controller based on improved active disturbance rejection control (ADRC). We propose the 3D simulation based on FlightGear to better reflect the state of quad tilt rotor in conversion mode. The simulation results show that the flight control system is effective. The 3D real scene simulation can directly observe the status of every control mechanism, which can provide a better and more intuitive analysis for manipulation strategy.

1. Introduction
The quad tilt rotor (QTR) combines the advantages of multi rotor and fixed wing with good vertical takeoff/landing and cruising performance. It has the characteristics of long range and high cruising efficiency. Because of its multi rotor and fixed wing UAV flight characteristics, the design of its flight control system is the focus and difficulty [1-7].

Papachristos [8] exhibited the nonlinear/linearized dynamics and the corresponding attitude tracking controller designed by PID controller. However, due to the nonlinear coupling of longitude and latitude manipulation and the indeterminate complex variable mechanical structure with tilting of the nacelle [9, 10], the control effect of the PID controller is limited. Oner [11] designed a LQR controller for position control of the vehicle in vertical flight mode. The flight control method based on the optimal control needs to obtain an accurate linear model by linearizing the nonlinear model. However, for the nonlinear model, the control range of the controller is limited to the vicinity of the trimming point. References [12-14] combined intelligent algorithms with classical control algorithms, considering that the tilt quad rotor is high order, time varying and nonlinear system in the transition mode, Cheng Peng [15] introduced neural network into PID controller to improve controller robustness. But the controller structure is more complicated and the design process is difficult.

In this paper we design an attitude controller based on active disturbance rejection control (ADRC). We propose a 3D simulation based on FlightGear and design a 3D model of quad tilt rotor. The 3D simulation is carried out to verify the model and flight control system.
2. Design of flight control law

Flight control law is the core of the flight control system. ADRC technology does not depend on the model [20-22], which can handle various internal uncertainties, and has strong robustness. ADRC consists of three parts: TD, ESO and NLSEF. The structure of active disturbance rejection controller is shown in Fig 1.

\[ \begin{align*}
    r_1(k+1) &= r_1(k) + h z_2(k) \\
    r_2(k+1) &= r_2(k) + h f_{st}(r_1(k) - v(k), r_2(k), \delta, h)
\end{align*} \]  

(1)

Where, \( r_1(k) \) is tracking signal of \( v(k) \), \( r_2(k) \) is the differential of \( \dot{v}(k) \), \( \delta \) is the parameter that determines the speed of tracking, \( h \) is step size. The nonlinear function \( f_{st}(x_1, x_2, \delta, h) \) expression is as follows:

\[ f_{st}(x_1, x_2, \delta, h) = \begin{cases}
    \delta \text{sign}(a), & |a| > d \\
    \delta a, & |a| \leq d
\end{cases} \]

\[ a = \begin{cases}
    x_1 + \left( \frac{a_0 - d}{2} \right) \text{sgn}(y), & |y| > d_0 \\
    x_1 + \frac{y}{h}, & |y| \leq d_0
\end{cases} \]  

\[ d = \delta h, d_0 = \delta h d, y = x_1 + hx_2, a_0 = \sqrt{d^2 + 8\delta} \]  

(2)

2.1. Tracking Differentiator

TD is tracking differentiator that arranges a transition process for the attitude angle command and obtains its differential signal, that is, the attitude angular rate command:

\[ \begin{align*}
    r_1(k+1) &= r_1(k) + h z_2(k) \\
    r_2(k+1) &= r_2(k) + h f_{st}(r_1(k) - v(k), r_2(k), \delta, h)
\end{align*} \]  

Where, \( r_1(k) \) is tracking signal of \( v(k) \), \( r_2(k) \) is the differential of \( \dot{v}(k) \), \( \delta \) is the parameter that determines the speed of tracking, \( h \) is step size. The nonlinear function \( f_{st}(x_1, x_2, \delta, h) \) expression is as follows:

\[ f_{st}(x_1, x_2, \delta, h) = \begin{cases}
    \delta \text{sign}(a), & |a| > d \\
    \delta a, & |a| \leq d
\end{cases} \]

\[ a = \begin{cases}
    x_1 + \left( \frac{a_0 - d}{2} \right) \text{sgn}(y), & |y| > d_0 \\
    x_1 + \frac{y}{h}, & |y| \leq d_0
\end{cases} \]  

\[ d = \delta h, d_0 = \delta h d, y = x_1 + hx_2, a_0 = \sqrt{d^2 + 8\delta} \]  

(2)

2.2. Extended State Observer

The ESO can simultaneously estimate the system states and total external disturbance. The structure of the ESO can be described as:

\[ \begin{align*}
    e &= z_1 - y \\
    z_1 &= z_2 - \beta_1 e \\
    z_2 &= z_3 - \beta_2 f_{al}(e, \alpha_1, \delta) + bu \\
    \bar{z}_2 &= -\beta_2 f_{al}(e, \alpha_1, \delta)
\end{align*} \]  

(3)

Where, \( \beta_i > 0 (i = 1, 2, 3) \), \( \alpha_1 = 0.5, \alpha_2 = 0.25 \). The function of saturation function \( f_{al}(e, \alpha, \delta) \) is to suppress signal jitter.

\[ f_{al}(e, \alpha, \delta) = \begin{cases}
    \frac{e}{\delta^2}, & |e| \leq \delta \\
    |e| \text{sgn}(e), & |e| > \delta
\end{cases} \]  

(4)
2.3. Nonlinear state error feedback
NLSEF is a nonlinear state error feedback that non-linearly combines the deviation between the output of the tracking differentiator and the state of the system. The traditional NLSEF control law of ADRC is as follows:

\[
\begin{align*}
    e_1 &= v_1 - z_1 \\
    e_2 &= v_2 - z_2 \\
    u &= \beta_1 f_a(e_1, \alpha_1, \delta) + \beta_2 f_a(e_2, \alpha_2, \delta)
\end{align*}
\]

Where, \(0 < \alpha_1 < 1 < \alpha_2\), \(k_p = \beta_1\), \(k_d = \beta_2\), \(e_1\) is error between the command signal and the controlled object position output, \(e_2\) is the error between the command differential signal and the controlled object speed output.

3. 3D Model of QTR

3.1. Design of 3D Model for QTR

Figure 2. 3D model of QTR

Figure 3. Documents of model
In FlightGear, the creation of a new aircraft can be roughly divided into four steps: create the 3D model of the aircraft, create the flight dynamics model of the aircraft, create the animation and the operation of the aircraft, and realize each subsystem of the aircraft. Specifically, it can be divided into two major development areas: flight dynamics model development and aircraft 3D model development.

Through OpenSceneGraph technology, FlightGear supports various 3D file formats, including VRML1, AC3D, DXF, etc. Among them, .ac file is the standard used in most FG models. The model development in this paper also uses AC3D for secondary development.

Figure 2 is the rendering of the quad tilt rotor model developed with AC3D software. During modeling, it is important to note that the definition of the object name corresponds to the name in the configuration file below. The property ‘/sim/model/path’ in the main FlightGear property tree controls which models will be loaded. The easiest way to load a new model is to use the ‘--prop’ command:

```
fgfs --prop:/sim/model/path=Models/V44.ac
```

If set the 3D model to the default value for the aircraft instead of specifying it on the command line, it is needed to edit the aircraft configuration file. When starting FlightGear with the ‘--aircraft’ option, it reads properties from one of the folders, such as:

```
fgfs --aircraft=V44
```

There are a series of configuration files in the aircraft model file, as shown in figure 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<PropertyList>
  <description>Quad-TiltRotor frans</description>
  <author>Zhijiang Wang</author>
  <configuration>Quad-TiltRotor</configuration>
  <version>3.0</version>
  <startup>
    <splash-texture> Aircraft/V44-TiltRotor/v44-splash.png</splash-texture>
  </startup>
  <sound>
    <path>Aircraft/V44-TiltRotor/Sounds/v44-sound.xml</path>
  </sound>
  <model>
    <path archive="y">/Aircraft/V44-TiltRotor/Models/v44.xml</path>
    <file type="string">default-file</file>
  </model>
  <entry>
    <model type="int">0</model>
    <wing state="float">0</wing state>
    <blade folding type="float">0</blade folding>
    <wing rotation type="float">0</wing rotation>
    <tilt type="float">0</tilt>
    <animation tilt type="float">0</animation tilt>
    <inputflag type="float">0</inputflag>
    <inputflag tilt type="float">0</inputflag tilt>
    <ringfieldincidence type="float">0</ringfieldincidence>
    <wing>
      <elevator type="float">0</elevator>
      <ailerons type="float">0</ailerons>
      <rudder type="float">0</rudder>
      <flap type="float">0</flap>
    </wing>
    <ref>
      <data>
        <elevator type="float">0</elevator>
        <collective type="float">0</collective>
      </data>
    </ref>
  </entry>
</PropertyList>
```

**Figure 4.** Code of the configuration

**Figure 5.** Code of the animation
Among them -- set.xml file is the launch default file, which is an important file describing the aircraft dependency. Set the path to a .xml file in this file. This file links to every other model file the aircraft needs. The partial code of the configuration file is shown in figure 4.

The Models folder contains all model-related files, such as textures, 3D models.ac, and animation.xml files. The entire aircraft can be modeled as a file or split up to make it easier to move parts of the aircraft. The partial code of the animation file is shown in figure 5.

The .XML in the home directory contains the entire flight dynamics model of the aircraft. There are three different flight dynamics models in FlightGear: YASim, JSBSim and UIUC. They have their own advantages and disadvantages. When model data is limited, YASim is considered the best way to model, and when real wind tunnel data is available, JSBSim is the best choice. All of them can be found in the open source model. The partial code of the dynamic model is shown in figure 6.

The Systems folder is system interaction files, including buttons and help documents. Sounds folder is a sound file for the actions of various parts of the model. The sound is very important for the actual feeling of the simulation.

### 3.2. 3D Real Scene Simulation Environment

This simulation environment is built in the ununtu 18.04 system. It is relatively easy to configure the environment in the Linux system with only a few lines of commands, and it is faster to develop under Linux than under Windows. First, install git environment through the following commands:

```
sudo apt-get update
sudo apt-get install git
sudo apt-get install ggit git-gui
```

When git is installed, clone the source from GitHub with the following commands:

```
git clone https://github.com/ArduPilot/ardupilot
cd ardupilot
git submodule update --init --recursive
```

Then install the build tool chain and configure the environment:

```
Tools/environment_install/install-prereqs-ubuntu.sh -y
~/.profile
```

Install and update MAVProxy and pymavlink:

```
```
pip install --upgrade pymavlink MAVProxy --user

Finally, FlightGear is installed to achieve 3D display:

```bash
sudo apt-get install flightgear
```

The environment for 3D real scene simulation is completed. Its architecture is shown in figure 7.

![Figure 7. The architecture of simulation](image)

For the source code of quad tilt rotor, we adopt the open source code based on Ardupilot for secondary development, which eliminates the preparation of the underlying driver and communication protocol. The partial code of quad tilt rotor is shown in figure 8.

![Figure 8. Partial code of quad tilt rotor](image)
4. Simulation Results
To start FlightGear, open the script file with the following command:

```
fg_quadplane_view.sh
```

When FlightGear starts, start the simulation environment with the following command:

```
sim_vehicle.py -v ArduPlane -f quadplane -L KSFO --console --map
```

After the simulation environment is started, commands can be entered through the terminal or directly through the ground station for simulation.

(a) Target tracking  (b) Target commands  (c) Path tracking  (d) Path commands

**Figure 9.** Commands and tracking results

(a) Helicopter mode  (b) Conversion mode  (c) Fixed wing mode  (d) Landing

**Figure 10.** Flight modes of quad tilt rotor in 3D simulation

In the process of simulation, first to takeoff in helicopter mode, as shown in figure 9(a) and (b), then set a target, let the quad tilt rotor fly to the target point automatically. In the process of flight, if the current flight speed reached the set value, the flight mode automatically from helicopter turn to fixed wing. When the quad tilt rotor reaches the target point, it will hover around the target position. Then the flight path planning is carried out for the quad tilt rotor. As shown in figure 9(c) and (d), the quad tilt rotor follows the set trajectory in fixed-wing mode and land in helicopter mode. Figure 10 shows the flight modes of quad tilt rotor during the whole flight. In the whole simulation process, the quad tilt rotor can follow the commands very well, and through the 3D real scene simulation, the nacelle state of the quad tilt rotor during the conversion mode and the flight state of the quad tilt rotor can be better reflected.

5. Conclusion
In this paper, we carry out the design of the controller based on improved active disturbance rejection control. We propose the 3D simulation based on FlightGear to verify the flight control system. The simulation results show that the flight control system is reasonable and effective. The 3D real scene simulation can directly observe the nacelle state and the flight state of the quad tilt rotor.

Acknowledgments
This work was financially supported by A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References
[1] Marr RL, Blackman S and Weiberg JA. Wind tunnel and flight test of the XV-15 Tilt Rotor Research Aircraft. J Aircr 1979; 19: 10051011.
[2] Wang Q and Wu W. Modelling and analysis of tiltrotor aircraft for flight control design. Inf Technol J 2014; 13: 855–894.

[3] Chun-hua LI and Guo-hua XU. Modeling and analytical investigation on rotor/wing aerodynamic interaction for tiltrotor aircraft. Acta Aerodyn Sin 2008; 3: 109–117.

[4] Guo J, Song Y and Xia P. Full envelope flight control method for small unmanned tilt rotor aircraft. J Nanj Univ Aeronaut Astronaut 2009; 41: 439–444.

[5] Song Y. Design of flight control system for a small unmanned tilt rotor aircraft. Chin J Aeronaut 2009; 22: 250–256.

[6] Li H, Qu X and Wang W. Multi-body motion modeling and simulation for tilt rotor aircraft. Chin J Aeronaut 2010; 23: 415–422.

[7] Rysdyk RT and Calise AJ. Adaptive model inversion flight control for tilt-rotor aircraft. J Guid Control Dyn 2012; 22: 402–407.

[8] Papachristos C, Alexis K and Tzes A. Design and experimental attitude control of an unmanned tiltrotor aerial vehicle. In: International conference on advanced robotics, 2011. Piscataway: IEEE.

[9] Sato M and Muraoka K. Flight controller design and demonstration of quad-tilt-wing unmanned aerial vehicle. J Guid Control Dyn 2015; 38: 1–12.

[10] Oner KT, Oner KT and C¸ etinsoy E. Modeling and position control of a new quad-rotor unmanned aerial vehicle with tilt-wing mechanism. International Conference on Control, Automation, Robotics and Vision (ICCARV’08), Laval, France. 2008.

[11] Oner KT, Cetinsoy E and Unel M. Dynamic model and control of a new quadrotor unmanned aerial vehicle with tilt-wing mechanism. In: Proceedings of World Academy of Science Engineering & Technology, No. 58, 2008. 2(9) 1008-1013.

[12] Flores G, Lugo I and Lozano R. 6-DOF hovering controller design of the Quad Tiltrotor aircraft: simulations and experiments. In: Decision & control, 2014, pp.6123, 6128. Piscataway: IEEE.

[13] Niu ZG and Zhang JM. Method of smooth-switch fuzzy PID for linear motor control. Proc CSEE 2006; 26: 132–136.

[14] Teng W, Pan H and Ren J. Neural network PID decoupling control based on chaos particle swarm optimization. In: Control conference, 2014, pp.5017, 5020. Piscataway: IEEE.

[15] Peng C, Wang XM and Chen X. Design of tiltrotor flight control system in conversion mode using improved neutral network PID. Adv Mat Res 2014; 850–851: 640–643.

[16] Yang Z, Rui H, Zhao Y, et al. Design of an Active Disturbance Rejection Control for Transonic Flutter Suppression. Journal of Guidance Control & Dynamics. 2017, 40(1): 1-12.

[17] Hua X, Yi Q, Fan G L, et al. Anti-crosswind Autolanding of UAVs based on Active Disturbance Rejection Control. 2013.

[18] Song S, Wang W, Lu K, et al. Nonlinear attitude control using extended state observer for tiltrotor aircraft. 2015.