Problem Chosen	2023 ShuWei Cup Summary Sheet	Team Control Number
A		2023111321427

Modeling and Optimization Control Issues of Compound Helicopter

Summary

In this study, the attitude control problem under different flight conditions is deeply discussed by combining theoretical analysis and numerical simulation for the flight control problem of the compound helicopter.

For question one, we expressed the pitch moment of the compound helicopter, considering its coaxial rigid rotors, propeller thrusters and horizontal tails. The acceleration of the pitch angle was derived by the **Newton-Euler's equation**. Pitch angles were further derived by linear integral, with **Runge-Kutta method** applied to give numerical solutions of **-0.1900, -0.7573 and -3.3035 radians at 5s, 10s and 20s** respectively.

For question two, we gave a holistic consideration on moments of three attitude angles. The rotor moment and yaw moment were both generated by coaxial rigid rotors and vertical tails, while the pitch moment took three components demonstrated in question one into consideration. Based on the respective moment, numerical solutions were given, where at 5s, 10s, and 20s, **roll angles** reached 1.5013, 5.9814 and 23.9736 radians, **pitch angles** were -0.1900, -0.7572 and -3.0351 radians, as well as **yaw angles** achieved -0.1857, -0.7388 and -2.9657 radians.

For question 3, with established models quantifying relationships between maneuvering characteristics and attitude angles, we reversely sought for maneuvers that **minimized the sum of absolute values** of attitude angles by the sequential least squares programming (SLSQP) method. Characteristics of two speed mode were successfully given.

For question 4, based on gradient descent and the model established before, we dynamically optimized the attitude angles **every 0.5 seconds** to reflect on the accelerated forward component of flight speed, in order to achieve level flight. The dynamic values of each control input show that our method significantly reduce the attitude angle deviation , maintain the high stability of the aircraft during the 20-second flight as well as exhibit a superior performance in coping with speed changes in dynamic environments.

In summary, we established the model to precisely control the attitude angles of a compound helicopter, which had important reference value for improving the flight stability and safety of helicopter.

Key word: Newton-Euler equations, dynamical models, Runge-Kutta methods, gradient descent, dynamic optimization adaptive control strategies

Content

1. Introduction

1.1 Background

Helicopters are versatile aircraft with the ability to perform tasks like vertical takeoff and landing, which makes them well-suited for various applications such as reconnaissance and transportation. However, since the rotor blades of traditional helicopters are easily affected by short waves in high-speed flight, it is thus difficult to guarantee its stability. To retain their high-speed flight capability, compound helicopters combining fixed wings with rotors has become a kind of solutions, where coaxial helicopters stands out as a typical one. Coaxial helicopters are featured with four components: coaxial rigid rotors, propeller thrusters, horizontal and vertical tails, of which the center of the last three components are all located on the fuselage's symmetry plane. These four power components significantly affect the corresponding moments, **roll moment**, **pitch moment**, and **yaw moment** of the flying machine's attitude angles.

- Coaxial rigid rotors generates the **three** aerodynamic moment, which can be proportional to the air density ρ, rotor disc area, and rotor blade tip velocity, with the positive scale factor called the rotor moment factor.
- The propeller thrusters generate thrust and rotational moment through rotation.
- The horizontal tails provide **pitch moment**, which is proportional to dynamic pressure, horizontal tail area, and the lateral distance between the horizontal tail and the center of mass, with the horizontal tail moment coefficient.
- Similarly, the vertical tails provide **yaw/roll moment**, which is proportional to dynamic pressure, vertical tail area, and the lateral/longitudinal distance between the vertical tail and the center of mass, with the vertical tail moment coefficient.

Generally, two modes depending on the flight speed need to be considered. The low-speed mode refers to speeds below 85 meters/second, of which the attitude angle is mainly controlled by coaxial rotor and propeller thrusters. The high-speed mode refers to speeds above 100 meters/second, of which attitude angle control is mainly realized through propeller thrusters, horizontal and vertical tails.

1.2 Work

Given the seven maneuverable parts of a composite helicopter, namely coaxial rotor overall distanceu_c, differential total distance u_{cd}, longitudinal cycle pitch u_e and lateral cycle pitch u_a, as well as propeller thruster operating capacity u_t, elevator deflection values u_{eh}, and rudder deflection valuesu_{av}, models for different case shall be established for each issue.

First, we need to model the pitch angle variation of a compound helicopter, by establishing the expression for **the pitch moment**, and provide the **attitude angles** at 5s, 10s, 20s, with an

initial flight altitude of $3,000$ meters, flight speed forward component $80m/s$, vertical ascent component $2m/s$, $u_c = 0$ degree, $u_{cd} = -2.1552$ degree, $u_e = -3.4817$ degree, $u_a = -2.0743$ degree, $u_t = 0$ degree, $u_{eh} = -9.0772 \times 10^{-7}$ degree, and $u_{av} = 4.1869 \times 10^{-7}$ degree.

Further, we aim to build models for the variation of attitude angles, by expressing **the roll, pitch and yaw moments** of the compound helicopter, and also provide the **attitude angles** of the flying machine at 5s, 10s, and 20s, with an initial flight altitude of $3,000$ meters, flight speed forward component $80m/s$, vertical ascent component $0.2m/s$, and the same maneuverable parameters given in question one.

For issue 3, the **magnitude of the maneuvers of each component** to satisfy the vehicle's level-flying mission (zero attitude angle) for the aircraft needs to be designed in **different speed mode**. (initial flight altitude of 3,000 meters, flight speed forward component of 80 meters/second, vertical ascent component of 0.2 meters/second; initial flight altitude of 3000 meters, flight speed forward component of 180 meters/second, vertical ascent component of 0.2 meters/second).

Finally, assuming the forward acceleration is provided solely by the propeller booster, when the flight speed forward component increases uniformly from 80 meters/second to 180 meters/second within 20 seconds, our aim is to design the **dynamic values of each control input** to achieve both forward acceleration and level flight (zero attitude angle) for the aircraft, while taking into account the maneuvering characteristics of the helicopter during low-speed and high-speed flight (initial flight altitude of 3,000 meters, flight vertical ascent component rate of 0.2 meters/second).

2. Problem analysis

2.1 Data Analysis

The data has given the basic parameters of a flying machine, such as the mass, the two tails' surface area, and the lateral and longitudinal distance from the center of mass to the propeller thruster and two tails. Meanwhile, those given maneuvering volumes can be used to get the moment coefficient of each attitude angles for each component, according to the corresponding values of **rotor advance ratio**.

2.2 Analysis of question one

In the low-speed mode of question one, where the flight speed forward component is 80 m/s, the attitude angle is realized mainly by coaxial rotor and propeller thrusters. However, the pitch moment of the compound helicopter may also be affected by the horizontal tails, like what we listed in the section 1.1.

So, the expression for the pitch moment here would mainly consist of coaxial rotor and propeller thrusters, with a consideration of the horizontal tails. By the calculation of the total pitch moment, a further model developed for the pitch angle variation would be based on the Newton-Euler's equation in rigid body dynamics, to obtain the gradients and accelerations, so as to provide the pitch angles at different timestamps.

2.3 Analysis of question two

Based on the preparations in question one, similar to the pitch moment, roll moment and yaw moment of the compound helicopter, provided by coaxial rigid rotors and vertical tails, also requires definition. With the calculated moment and given axis moment of inertia, the acceleration of each attitude angle could be derived by Euler's rotation equations. The attitude angles could then be obtained by linearly integrating twice.

2.4 Analysis of question three

The first two questions have given seven fixed maneuvering characteristics to figure out the attitude angles, whereas in return, question three seeks for those maneuvering characteristics to achieve zero attitude angles in two speed mode. The relationship between attitude angles and maneuvering characteristics have been modeled before. One thing needs to be done here is to minimize the sum of the absolute values of three angles within the seven given ranges. In the ideal level-flying mission, values of the seven characteristics at random timestamp that achieve the minimum sum, are definitely the anticipated results.

2.5 Analysis of question four

Question four designs a more complex situation, where the flight speed forward component increases uniformly from 80 meters/second to 180 meters/second within 20 seconds, with an assumption that the forward acceleration will be provided solely by the propeller booster). Based on the zero-attitude-angel model of the third questions, we need to further take the forward acceleration into account, which indicates the rotor advance ratio that decides parameters in the calculation of rotor moments and yaw moments is not immobile. Its discrete distribution then inevitably calls approximations in the practical usage. Within the acceleration period of 20s, we need to dynamically obtain the numerical solution of maneuvering characteristics like question three, so a smaller interval of 0.5s is applied to monitor 40 timestamps in the record of dynamic values of each control input.

3. Symbols and Analysis

3.1 Symbol Description

Table 1 Descriptions of symbols

Symbols	Descriptions
M	Moment
F	Force
d	Distance
C	Moment coefficients
Q	Propeller thruster rotational torque
I	Moment of Inertia

3.2 Fundamental assumptions

1. The four components could all be viewed as rigid body and the elastic deformation of the fuselage is not considered.
2. The flying machine would not be affected by the air disturbance.
3. In the level-flying task, aircraft would only be affected by the given maneuvering characteristics.
4. The altitude could be approximated at the height of 3,000 meters, with a constant air density and without the consideration of compression characteristics of air.
5. The ground coordinate system is regarded as an inertial coordinate system.
6. Ignoring the curvature of the earth.
7. The mass distribution of each part of the sample composite high-speed helicopter is constant and does not change with time.
8. Ignoring the dynamic characteristics of the power system and actuator.

4. Question one

4.1 Models

The modeling process has been proposed in the analysis above. Specifically, two steps were taken, which were the calculation of total pitch moment $M_{totalpitch}$ in Eq.(1) and the modeling

of pitch angle variation, where the $M_{totalpitch}$ is proportional to the air density ρ, rotor disc area A, and rotor blade tip velocity V_{tip}, as well as the rotor pitch moment coefficient $C_{rotorpitch}$, calculated by Eq.(2). Apart from u_e directly given in the data, the rest of parameters in $C_{rotorpitch}$ could be found according to the *rotor advance ratio*, which is calculated by $flightspeed\ /\ V_{tip}$. In question one, the rotor advance ratio is 0.0740, approximating to 0.1,

$$M_{total\ pitch} = M_{rollpitch} + M_{propellerpitch} + M_{horizontalpitch} \tag{1}$$

$$\begin{aligned} C_{rotor\ pitch} =& pitch\ deviation\ value + \\ & longitudinal\ variable\ pitch\ coefficient \times u_e + \\ & total\ pitch\ coefficient \times overall\ distance + \\ & differential\ total\ pitch\ coefficient \times \\ & differential\ total\ distance \end{aligned} \tag{2}$$

In general, the moment could all be calculated by Eq.(3), where F represents the force applied, and d describes the distance from the axis of rotation to the point where the force is applied. since α is the angle between the lever arm and the force vector, in this paper $sin(\alpha)$ equals to 1. As a result, the three moments could all follow the basic formula.

$$M = F \times d \times sin(\alpha) \tag{3}$$

Concretely, the pitch moment of coaxial rigid rotors $M_{rotorpitch}$ could be given by Eq.(4), where R is the radius of rotors, namely the distance, and $q = \frac{1}{2}\rho V_{tip}^2$ denotes the dynamic pressure, forming the force with the area A. It is worth to mention that V_{tip} is calculated by $coaxial\ rigid\ rotor\ speed \times R$. Besides, although the aircraft holds a vertical ascent component of 2m/s, within 20s, the increment of 40 meters, increasing from 3,000 meters to 3,040 meters, could be ignored. Therefore, we approximates ρ at the height of 3,000 meters to be $0.9093 kg/m^3$[1].

$$M_{rotor\ pitch} = C_{rotor\ pitch} \times qA \times R \tag{4}$$

Meanwhile, the pitch moment of propeller thrusters consists of thruster and rotational moment Eq.(5), where $d_propeller$ represents the longitudinal distance from center of mass to propeller thruster above the fuselage, and Q describes the propeller thruster rotational torque, to be 0 when u_t equals to 0 in question 1.

$$M_{propell\ pitch} = d_{propeller} \times u_t + Q \tag{5}$$

As for the pitch moment of the horizontal tails $M_{horizontal\ tail\ pitch}$ in Eq.(6), $d_{horizontal\ tail}$ is the lateral distance from center of mass to horizontal tail, and $S_{horizontal\ tail}$ gives the value of horizontal tail surface area.

$$M_{horizontal\ pitch} = C_{horizontal\ tail} \times q \times S_{horizontal\ tail} \times d_{horizontal\ tail} \tag{6}$$

The variation of pitch angles is not only affected by moments of coaxial rigid rotors, propeller thrusters and horizontal tails, but is also influenced by the mass of the helicopter and three axes moment of inertia. So, in the rotation equations of rigid body dynamics, the variation of pitch angles could be obtained by calculating the its acceleration θ'' in Eq.(7)[2].

$$I_{pitch}\theta'' = M_{total\ pitch} \tag{7}$$

In addition, the gradients of pitch angle θ' is the linear integral of θ'' over time, and could be represented by $\theta' = \int \theta'' dt$, where we could get the value of pitch angle by a further integral of θ' over time.

4.2 Results

In this question, our work consists of four important variables, which can be seen in Appendix 9..

- Total pitch moment of coaxial rigid rotors, propeller thrusters, where we further take horizontal tails into consideration.
- Acceleration of pitch angle derived by total pitch moment and moment of inertia.
- Gradients of pitch angles linearly integrated by the acceleration values over time.
- The final pitch angle further integrated by the gradients over time.

We applied the odeint module with the Runge-Kutta method in Python [3] to get numerical solutions at 5s, 10s, and 20s (Table.2), and there was only a little difference after the addition of horizontal tails, which aligns with the main influence of coaxial rigid rotors and propeller thrusters. For a comprehensive consideration, we made the results of three components as our final answer.

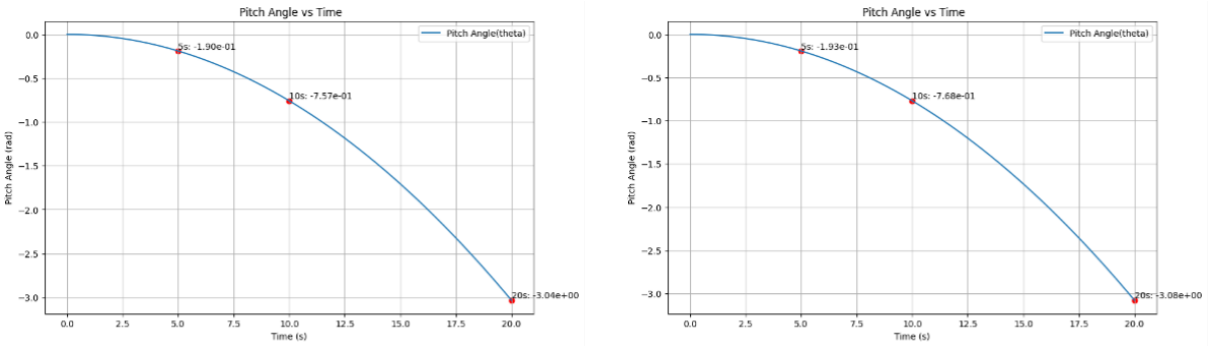
From these data (Figure.1), it can be observed that the pitch angle of the aircraft gradually decreases (becomes negative) over time, indicating that the nose of the aircraft is tilted downward relative to the horizontal plane, and this tilt increases along with time. The curve of the pitch angle shows the drop of the pitch angle over time is nonlinear, and the descent rate increases with time. This trend may be due to the combined effects of the aerodynamic and propeller moments of the aircraft.

From these figures, it could be seen that as time passes by, the attitude angles of the aircraft all show a clear growth trend. In particular, the roll angle φ increases at the most significant rate, indicating that the roll dynamics of the aircraft are the most active under these conditions. In contrast, the pitch angle θ and yaw angle ϕ show a slower but still a steady upward trend.

These results are essential for understanding and predicting the dynamic behavior of compound helicopters under specific flight conditions. By analyzing these changes in attitude angles, we can provide an important theoretical basis for the design of the control system of the aircraft,

Table 2 Pitch angles in different cases

	With horizontal tails	Without horizontal tails
5s	-0.1900	-0.1928
10s	-0.7573	-0.7683
20s	-3.3035	-3.0793



(a) with horizontal tails (b) without horizontal tails

Figure 1 Pitch angles of the compound helicopters at three timestamps

so as to improve the stability and maneuverability of the aircraft. In addition, these analyses help aircraft designers understand the response characteristics of the aircraft under different maneuvering inputs, and provide a reference for optimizing the aircraft design. It is important to note that these results are based on idealized assumptions, and the behavior of the actual aircraft may be affected by a variety of factors, such as airflow disturbances, structural elastic deformation, etc. Therefore, in practical applications, these theoretical predictions need to be combined with experimental data and flight test results to obtain a more comprehensive and accurate analysis of aircraft performance.

5. Question two

5.1 Models

Question 2 requires the expressions of not only pitch moment, but also roll and yaw moment in a low-speed mode similar to question one. The vertical ascent component here is 0.2m/s. Considering the little difference of altitudes within 20s, the air density applied here and the following same conditions remains $0.9093 kg/m^3$. Based on the model above, the composition of each type of moment is listed as follows.

- roll moments M_{roll} and yaw moments M_{yaw} are both produced by coaxial rigid rotors and vertical tails.
- pitch moments consist of coaxial rigid rotors, propeller thrusters, with horizontal tails of necessity, analyzed in section 4.2.

As a consequence, M_{roll} could be given by Eq.(8). Like what we discussed above, these moments are all composed of the multiplication of force applied ($\frac{1}{2}\rho V_{tip}^2 \times area$)and perpendicular distance (d). So, for coaxial rotors, area equals to πR^2, and d denotes R, while for vertical tails, area equals to vertical tail surface area, and d denotes the **lateral** distance from center of mass to vertical tail. M_{yaw} could be given Eq.(9), where the only difference from roll moment is d in the calculation of $M_{vertical\ tail\ yaw}$ refers to the **longitudinal** distance from center of mass to vertical tail

$$M_{total\ roll} = M_{rotor\ roll} + M_{vertical\ tail\ roll} \tag{8}$$

$$M_{total\ yaw} = M_{rotor\ yaw} + M_{vertical\ tail\ yaw} \tag{9}$$

$$M = C\frac{1}{2}\rho V_{tip}^2 \times area \times d \tag{10}$$

For C of each moment, $C_{rotor\ roll}$ is calculated by roll deviation value + lateral pitch roll factor $\times\ u_a$ + differential total distance roll coefficient $\times\ u_{cd}$. $C_{vertical\ tail\ roll}$ is calculated by vertical tail moment coefficient deviation + rudder coefficient $\times\ u_{av}$; $C_{rotor\ yaw}$ is derived by yaw deviation + differential total pitch yaw coefficient $\times\ u_{cd}$, while $C_{vertical\ tail\ yaw}$ is identical to $C_{vertical\ tail\ roll}$.

With the same expression of $M_{total\ pitch}$ from section 4, the three moment $M_{total\ pitch}$, $M_{total\ roll}$ and $M_{total\ yaw}$ can now be passed to the dynamical equation Eq.(7) and Eq.(11) to get acceleration of pitch angle θ'', roll angle φ'', and yaw angle ψ'', where I represents the moment of inertia for each angle.

$$I_{roll}\varphi'' = M_{total\ roll} \tag{11}$$

$$I_{yaw}\psi'' = M_{total\ yaw} \tag{12}$$

5.2 Results

Based on the derivation above, since the three acceleration of attitude angles are linear along time, we applied linear integral twice to get the final numerical solution at timestamps of 5s, 10s and 20s. In the same way, odeint module in Python Appendix (9.) gave the results in Table.3.

Table 3 Attitude angles at different timestamps

	Roll angle	Pitch angle	Yaw angle
5s	1.5013	-0.1900	-0.1857
10s	5.9814	-0.7572	-0.7399
20s	23.9736	-3.0351	-2.9657

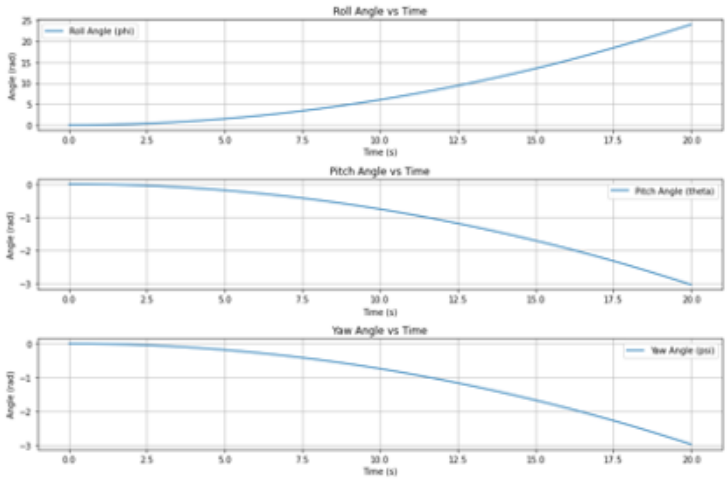


Figure 2 Attitude angles of the compound helicopters at three timestamps

6. Question three

6.1 Models

In question three, the high-speed mode and low-speed need to be addressed respectively, due to the different values based on the *rotor advance ratio*.

Models developed in the first two equations could be continuously applied, which means $M_{total\ rotor}$ and $M_{total\ yaw}$ are still calculated by Eq.(10). $M_{total\ pitch}$ is derived by Eq.(1). The difference is that in Eq.(5), the rotational torque will be considered in accordance with the propeller thruster operating capacity, u_t. Since the distribution of u_t in the given material is discrete, in the optimization, if u_t is out of the given values, then the $M_{propell\ pitch}$ still returns 0.

The objective of the process is to minimize the sum of absolute values of Eq.(13). We used Sequential Least Squares Programming (SLSQP) method [4] here to seek for values of seven variables that achieve the minimum values of f within the selectable ranges.

$$u_c, u_{cd}, u_e, u_a, u_t, u_{eh}, u_{av} = argmin(|\varphi| + |\theta| + |\phi|) \tag{13}$$

$$
\begin{aligned}
S.t :& u_c \in [0, 30], u_{cd} \in [-25, 25] \\
& u_e \in [-25, 25], u_a \in [-25, 25], \\
& u_t \in [0, 36], u_{eh} \in [-25, 25], u_{av} \in [-25, 25]
\end{aligned}
$$

6.2 Results

Rotor advance ratio in low-speed and high-speed mode was 0.074 and 0.067, approximating to 0.1 and 0.2 respectively. The parameters are settled down as Table.4

Table 4 Parameters at different mode

Parameters	low-speed mode	high-speed mode
Roll deflection value	0.00035	0.0004
Lateral pitch roll factor	0.00032	0.0004
Differential total distance roll coefficient	-0.00019	-0.00024
Pitch deviation value	0.0014	0.0029
Longitudinal variable pitch coefficient	0.00038	0.00045
Total pitch coefficient	0.00017	0.00034
Differential total pitch coefficient	0.00005	-0.000001
Yaw deviation value	0.0002	0.0001
Differential total pitch yaw coefficient	0.00011	0.00008
Horizontal tail moment coefficient deviation	-0.0001	-0.0001
Elevator coefficient	-0.00001	-0.00005
Vertical tail moment coefficient deviation	-0.0001	-0.00015
Rudder coefficient	-0.000005	-0.000016

The concrete variables and equations in each step are as listed follows, which can be seen in Appendix 9..

1 Definition of known parameters, such as axis moment of inertia, air density, etc.
2 Definition of three moments.
3 Modeling relationships between angles and moments.
4 Definition of objective function.
5 Numerical solution reached by SLSQP method, with initial values set as 0.

Considering the level-flying mission, ideally, the aircraft would remain the same maneuvering characteristics, which means that those characteristics are fixed at any timestamp. So, we randomly minimize the objective function at a certain timestamp, namely at 20s, with results shown in Table.5. We further validated our model by calculating the attitude angles based on obtained seven maneuvers. Figure.3 illustrated values of three angles, which significantly proved the efficacy of our model.

Table 5 Magnitude of the maneuvers at different mode

Maneuvering characteristics	low-speed	high-speed
u_c	0.4401	0.9161
u_{cd}	-1.8166	-1.2497
u_e	-3.6432	-7.1403
u_a	-2.1731	-1.7506
u_t	0	0
u_{eh}	-0.0009	-0.0093
u_{aav}	-0.0001	-0.0007

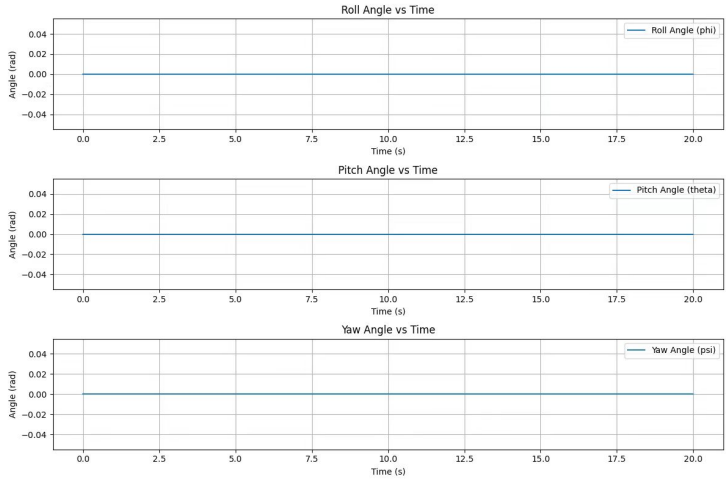


Figure 3 Validation of level flight

By analyzing the maneuvers of each component under low and high flight conditions, the following conclusions can be obtained:

At the forward flight speed of 80 m/s and vertical ascent component of 0.2 m/s, the total distance (u_c) and differential total distance (u_{cd}) of the coaxial rotors are 0.44 and -1.82 degrees, respectively, indicating that the attitude of the rotor needs to be moderately adjusted to maintain

flight balance.Negative values for longitudinal periodic pitch (u_e) and lateral periodic pitch (u_a) indicate that the rotor needs to be deflected in opposite directions to counteract instability in flight.The lower values for the elevator deflection (u_{eh}) and rudder deflection (u_{av}) indicate that the adjustment of these components is relatively small in low-speed flight.

At a forward flight speed of 180 m/s and a vertical ascent component of 0.2 m/s, the total distance (u_c) and total differential distance (u_{cd}) of the coaxial rotor have been adjusted to an increased extent, reflecting the need for greater rotor attitude adjustment to maintain stability in high-speed flight. In addition, the negative value of the longitudinal periodic pitch (u_e) further increases, indicating that the rotor needs to be adjusted more significantly to cope with the change of flight dynamics in high-speed flight. The amount of deflection of the elevator and rudder has also increased, indicating that these components play a more important role in maintaining flight stability at high speeds.

7. Question four

The modeling process is just similar to what we did in section 6.1 for both low-speed mode and high-speed mode, with only a little difference on the parameters, due to the dynamical values of *rotor advance ratio*. A smaller interval of 0.5s is used for generating more points in the optimization. At each point the *rotor advance ratio* was approximated to be 0.1, 0.2 or 0.3, in order to get the parameters like Table.4. Steps are as follows, which can be seen in Appendix 9...

1 Definition of given parameters, such as axis moment of inertia, air density, etc.
2 Definition of three moments.
3 Adding a dynamical map between *rotor advance ratio* and parameters
3 Modeling relationships between angles and moments.
4 Definition of objective function.
5 Numerical solution reached by SLSQP method, with initial values set as 0.

A total of 40 timestamps were recorded with results displayed in the Appendix 9.. Figure.4 gave a more direct illustration of the seven characteristics. Correspondingly, the variation of attitude angles within this acceleration period was shown in Figure.5.

The deviation values between average angles within 20s and the zero angles are $6.81 * 10^{-5} radian$ for the roll angle, $5.04 * 10^{-5} radian$ for the pitch angle, and $1.88 * 10^{-5} radian$ for the yaw angle. These extremely small mean deviation values indicate that the control strategy performs well in maintaining postural stability, and the our algorithm can effectively adjust the attitude of the aircraft to adapt to the change of speed.

The maximum values between average angles within 20s and the zero angles are $7.65 * 10^{-4} radian$ for the roll angle, $8.68 * 10^{-4} radian$ for the pitch angle, and $4.74 * 10^{-4} radian$ for

the yaw angle. Despite of the relatively large values compared to average ones, the maximum deviation is still within an acceptable range. These deviations may occur in the instance when the control parameters are adjusted, but they do not lead to significant fluctuations in the attitude of the aircraft, indicating that the control system has a certain degree of robustness.

The sum of the deviation value of the three angles are $2.72 * 10^{-3} radian$ for the roll angle, $2.01 * 10^{-3} radian$ for the pitch angle, and $0.07 * 10^{-3} radian$ for the yaw angle, which quantify the total effect of attitude control over the course of the flight. The low total deviation shows that the control algorithm maintains a good attitude control effect throughout the flight, and can ensure the stability of the aircraft's attitude even under dynamic conditions.

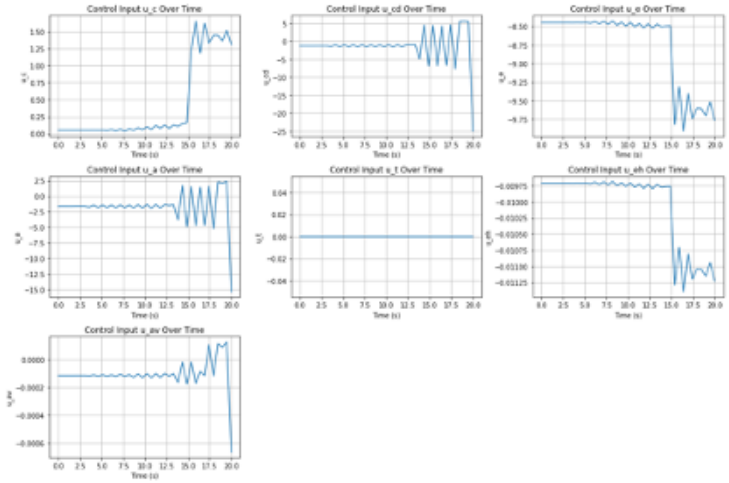


Figure 4 Dynamical maneuvers in 20s

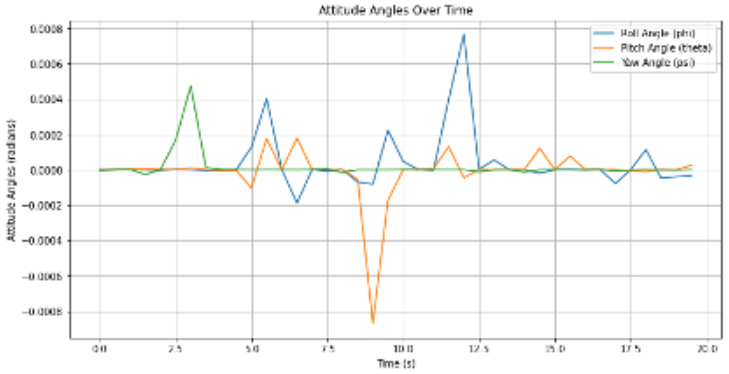


Figure 5 Variations of attitude angles within 20s

8. Pros and cons in our models

8.1 Pros

1 **Comprehensiveness and accuracy**. By comprehensively considering the moments generated by the coaxial rigid rotor, propeller thruster, and horizontal and vertical tails, this model comprehensively captures the dynamic characteristics of the composite helicopter and provides accurate description and control of the flight attitude.

2 **Strong adaptability**. The model successfully adapts to different flight conditions, such as low-speed and high-speed flight conditions, and can accurately predict and adjust the attitude of the aircraft in different flight stages, demonstrating good environmental adaptability.

3 **Efficient optimization control strategy**. Using the dynamic optimization adaptive control strategy based on gradient descent, the model can quickly respond to changes in aircraft speed and effectively maintain attitude stability, especially in the process of processing acceleration.

4 **Wide applicability**. The model provides a theoretical basis and practical scheme for the stable flight control of composite helicopters in practical applications, which is of great significance for improving aviation safety and flight efficiency.

8.2 Cons

1 **High computational complexity**. Considering multiple sources of torque and changes in flight conditions, the computational complexity involved in our model is relatively complex and requires high computational resources, which may limit its application in real-time or resource-constrained scenarios.

2 **Deviation between theory and practice**. Although the model is theoretically accurate, there may be deviations between the actual flight state and the model prediction due to environmental interference, mechanical errors and other factors in practical application.

3 **Sensitivity of model parameters**. The prediction accuracy of the model is highly dependent on the accuracy of the input parameters, such as rotor moment coefficient, air density, etc. Small changes in these parameters may have a significant impact on the final result.

4 **Balance between update frequency and real-time**. Choosing the right update frequency is a challenge in a dynamic control strategy. Excessive updates can overload the system, while insufficient updates can affect the real-time and accuracy of controls.

9. Further development

In this study, we have achieved preliminary results by using a dynamic optimization adaptive control strategy based on gradient descent for the attitude control task in the acceleration process of the composite helicopter. In order to further improve the efficiency and accuracy of control strategies, this section will explore the feasibility and potential advantages of combining genetic algorithms and artificial neural networks to solve question four.

1 **Application of Genetic Algorithm**. Genetic algorithm is an optimization algorithm that mimics the mechanism of biological evolution, which can efficiently find the global optimal solution in a wide search space through selection, crossover, and mutation. During helicopter acceleration, genetic algorithms can be used to optimize control parameters such as joystick angle, propeller thrust, etc. By setting the appropriate fitness function (such as minimizing the attitude angle deviation), the algorithm can automatically adjust the parameters to adapt to the change of flight speed and find the optimal control strategy to maintain the attitude stability.

2 **Integration of artificial neural networks**. Artificial neural networks have powerful data processing and pattern recognition capabilities, and are able to model complex nonlinear relationships. In this study, neural networks can be used to learn and predict the dynamic response of the aircraft. Through training, the network is able to identify the optimal control strategy in different flight states and adjust the control input in real time to maintain the optimal flight attitude.

3 **Algorithm Integration and Strategy Optimization**. Combining the advantages of genetic algorithms and artificial neural networks, we can create a powerful control system. Genetic algorithms play a role in the global search of control parameters, while neural networks are responsible for fine-tuning and prediction in these candidate solutions. This ensemble method can improve the search efficiency and resolution quality of the algorithm, and make the control strategy more accurate and flexible.

4 **Challenges and coping strategies**. Although the combination of genetic algorithms and artificial neural networks has significant advantages in theory, there are also some challenges in practical applications, such as the complexity of algorithms, the high demand for training data, and the requirements of real-time performance. To address these challenges, we can improve system performance by optimizing algorithm architecture, streamlining network models, and employing efficient parallel computing techniques. At the same time, by introducing the combination of simulation data and actual flight data, the generalization ability and practicability of the model can be improved.

5 **Future research directions**. Future research can focus on improving the real-time and robustness of the algorithm, as well as expanding the control strategy to cope with more

kinds of flight missions and environmental changes. In addition, further model validation and real-world flight tests are necessary steps to ensure the effectiveness and safety of the control strategy in real-world applications.

References

[1] https://www.engineeringtoolbox.com/standard-atmosphere-d_604.html

[2] Borisov A, Mamaev I S. Rigid body dynamics[M]. Walter de Gruyter GmbH and Co KG, 2018.

[3] E. Hairer, S.P. Norsett and G. Wanner, Solving Ordinary Differential Equations i. Nonstiff Problems. 2nd edition. Springer Series in Computational Mathematics, Springer-Verlag (1993).

[4] Carayannis G, Manolakis D, Kalouptsidis N. A fast sequential algorithm for least-squares filtering and prediction[J]. IEEE transactions on acoustics, speech, and signal processing, 1983, 31(6): 1394-1402.

Appendix A

Listing 1: The python source code for solving question one

```
from scipy.integrate import odeint
import numpy as np

I_pitch = 20000
rho = 0.9093
R = 6
A = np.pi * R**2
V_tip = 180
S_horizontal_tail = 1
d_horizontal_tail = -3
d_propeller = -0.2

u_c = 0
u_cd = -2.1552
u_e = -3.4817
u_a = -2.0743
u_t = 0
u_eh = -9.0772e-7
u_av = 4.1869e-7

flight_speed = 80
rotor_advance_ratio = flight_speed / V_tip

pitch_deviation_value = 0.0014
longitudinal_variable_pitch_coefficient = 0.00038
total_pitch_coefficient = 0.00017
differential_total_pitch_coefficient = 0.00005

horizontal_tail_moment_coefficient_deviation = -0.0001
elevator_coefficient = -0.00001

def calculate_pitch_moment(u_c, u_cd, u_e, u_eh, u_t):

    C_rotor_pitch = pitch_deviation_value +
        longitudinal_variable_pitch_coefficient * u_e + total_pitch_coefficient
        * u_c + differential_total_pitch_coefficient * u_cd
```

```
    M_rotor_pitch = C_rotor_pitch * 0.5 * rho * V_tip**2 * A * R
    C_horizontal_tail_pitch = horizontal_tail_moment_coefficient_deviation +
        elevator_coefficient * u_eh
    M_horizontal_tail_pitch = C_horizontal_tail_pitch * 0.5 * rho * V_tip**2 *
        S_horizontal_tail * d_horizontal_tail
    M_propeller_pitch = u_t * d_propeller
    M_pitch = M_rotor_pitch + M_propeller_pitch + M_horizontal_tail_pitch
    return M_pitch

def dynamics(y, t, I_pitch):
    theta, theta_dot = y
    M_pitch = calculate_pitch_moment(u_c, u_cd, u_e, u_eh, u_t)
    dtheta_dot_dt = M_pitch / I_pitch
    return [theta_dot, dtheta_dot_dt]

time_points = np.linspace(0, 20, 1000)

initial_conditions = [0, 0]

solution = odeint(dynamics, initial_conditions, time_points, args=(I_pitch,))

index_5s = np.argmin(np.abs(time_points - 5))
index_10s = np.argmin(np.abs(time_points - 10))
index_20s = np.argmin(np.abs(time_points - 20))

theta_5s = solution[index_5s, 0]
theta_10s = solution[index_10s, 0]
theta_20s = solution[index_20s, 0]

print(theta_5s, theta_10s, theta_20s)
```

Listing 2: The python source code for plotting question one

```
plt.figure(figsize=(10, 6))
plt.plot(time_points, solution[:, 0], label='Pitch Angle(theta)')

plt.scatter([5, 10, 20], [theta_5s, theta_10s, theta_20s], color='red')
plt.text(5, theta_5s, f'5s: {theta_5s:.2e}', verticalalignment='bottom')
plt.text(10, theta_10s, f'10s: {theta_10s:.2e}', verticalalignment='bottom')
plt.text(20, theta_20s, f'20s: {theta_20s:.2e}', verticalalignment='bottom')
```

```
plt.xlabel('Time (s)')
plt.ylabel('Pitch Angle (rad)')
plt.title(' Pitch Angle vs Time')
plt.legend()
plt.grid(True)
plt.show()
```

Appendix B

Listing 3: The python source code for solving question two

```
from scipy.integrate import odeint
import numpy as np

I_roll = 8000
I_pitch = 20000
I_yaw = 25000
rho = 0.9093
R = 6
A = np.pi * R**2
V_tip = 180
S_horizontal_tail = 1
d_horizontal_tail = -3
S_vertical_tail = 0.5
d_vertical_tail = -3
L_vertical_tail = 0.2
d_propeller = -0.2

u_c = 0
u_cd = -2.1552
u_e = -3.4817
u_a = -2.0743
u_t = 0
u_eh = -9.0772e-7
u_av = 4.1869e-7

flight_speed = 80
```

```
rotor_advance_ratio = flight_speed / V_tip

roll_deviation_value = 0.00035
lateral_pitch_roll_factor = 0.00032
differential_total_distance_roll_coefficient = -0.00019

pitch_deviation_value = 0.0014
longitudinal_variable_pitch_coefficient = 0.00038
total_pitch_coefficient = 0.00017
differential_total_pitch_coefficient = 0.00005

yaw_deviation_value = 0.0002
differential_total_pitch_yaw_coefficient = 0.00011

horizontal_tail_moment_coefficient_deviation = -0.0001
elevator_coefficient = -0.00001

vertical_tail_moment_coefficient_deviation = -0.0001
rudder_coefficient = -0.000005

def calculate_moments(u_c, u_cd, u_e, u_a, u_t, u_eh, u_av):

    C_rotor_roll = roll_deviation_value + lateral_pitch_roll_factor * u_a +
        differential_total_distance_roll_coefficient * u_cd
    M_rotor_roll = C_rotor_roll * 0.5 * rho * V_tip**2 * A * R
    C_vertical_tail_roll = vertical_tail_moment_coefficient_deviation +
        rudder_coefficient * u_av
    M_vertical_tail_roll = C_vertical_tail_roll * 0.5 * rho * V_tip**2 *
        S_vertical_tail * d_vertical_tail
    M_roll = M_rotor_roll + M_vertical_tail_roll

    C_rotor_pitch = pitch_deviation_value +
        longitudinal_variable_pitch_coefficient * u_e + total_pitch_coefficient
        * u_c + differential_total_pitch_coefficient * u_cd
    M_rotor_pitch = C_rotor_pitch * 0.5 * rho * V_tip**2 * A * R
    C_horizontal_tail_pitch = horizontal_tail_moment_coefficient_deviation +
        elevator_coefficient * u_eh
    M_horizontal_tail_pitch = C_horizontal_tail_pitch * 0.5 * rho * V_tip**2 *
        S_horizontal_tail * d_horizontal_tail
```

```
    M_propeller_pitch = u_t * d_propeller
    M_pitch = M_rotor_pitch + M_propeller_pitch + M_horizontal_tail_pitch

    C_rotor_yaw = yaw_deviation_value +
        differential_total_pitch_yaw_coefficient * u_cd
    M_rotor_yaw = C_rotor_yaw * 0.5 * rho * V_tip**2 * A * R
    C_vertical_tail_yaw = vertical_tail_moment_coefficient_deviation +
        rudder_coefficient * u_av
    M_vertical_tail_yaw = C_vertical_tail_yaw * 0.5 * rho * V_tip
    M_vertical_tail_yaw = C_vertical_tail_yaw * 0.5 * rho * V_tip**2 *
        S_vertical_tail * L_vertical_tail
    M_yaw = M_rotor_yaw + M_vertical_tail_yaw

    return M_roll, M_pitch, M_yaw

def dynamics(y, t, I_roll, I_pitch, I_yaw):
    phi, theta, psi, phi_dot, theta_dot, psi_dot = y
    M_roll, M_pitch, M_yaw = calculate_moments(u_c, u_cd, u_e, u_a, u_t, u_eh,
        u_av)
    dphi_dot_dt = M_roll / I_roll
    dtheta_dot_dt = M_pitch / I_pitch
    dpsi_dot_dt = M_yaw / I_yaw
    return [phi_dot, theta_dot, psi_dot, dphi_dot_dt, dtheta_dot_dt,
        dpsi_dot_dt]

time_points = np.linspace(0, 20, 1000)

initial_conditions = [0, 0, 0, 0, 0, 0]

solution = odeint(dynamics, initial_conditions, time_points, args=(I_roll,
    I_pitch, I_yaw))

index_5s = np.argmin(np.abs(time_points - 5))
index_10s = np.argmin(np.abs(time_points - 10))
index_20s = np.argmin(np.abs(time_points - 20))

phi_5s, theta_5s, psi_5s = solution[index_5s, :3]
phi_10s, theta_10s, psi_10s = solution[index_10s, :3]
phi_20s, theta_20s, psi_20s = solution[index_20s, :3]
```

```
print((phi_5s, theta_5s, psi_5s), (phi_10s, theta_10s, psi_10s), (phi_20s,
    theta_20s, psi_20s))
```

Listing 4: The python source code for plotting question two

```
time_points = np.linspace(0, 20, 1000)

initial_conditions = [0, 0, 0, 0, 0, 0]

solution = odeint(dynamics, initial_conditions, time_points, args=(I_roll,
    I_pitch, I_yaw))

phi, theta, psi = solution[:, 0], solution[:, 1], solution[:, 2]

plt.figure(figsize=(12, 8))

plt.subplot(3, 1, 1)
plt.plot(time_points, phi, label='Roll Angle (phi)')
plt.xlabel('Time (s)')
plt.ylabel('Angle (rad)')
plt.title('Roll Angle vs Time')
plt.grid(True)
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(time_points, theta, label='Pitch Angle (theta)')
plt.xlabel('Time (s)')
plt.ylabel('Angle (rad)')
plt.title('Pitch Angle vs Time')
plt.grid(True)
plt.legend()

plt.subplot(3, 1, 3)
plt.plot(time_points, psi, label='Yaw Angle (psi)')
plt.xlabel('Time (s)')
plt.ylabel('Angle (rad)')
plt.title('Yaw Angle vs Time')
plt.grid(True)
plt.legend()
```

```
plt.tight_layout()
plt.show()
```

Appendix C

Listing 5: The python source code for solving question three

```
from scipy.optimize import minimize
from scipy.integrate import odeint
import numpy as np

I_roll = 8000
I_pitch = 20000
I_yaw = 25000
rho = 0.9093
R = 6
A = np.pi * R**2
V_tip = 180
S_horizontal_tail = 1
d_horizontal_tail = -3
S_vertical_tail = 0.5
d_vertical_tail = -3
L_vertical_tail = 0.2
d_propeller = -0.2

flight_speed = 80
rotor_advance_ratio = flight_speed / V_tip

roll_deviation_value = 0.00035
lateral_pitch_roll_factor = 0.00032
differential_total_distance_roll_coefficient = -0.00019
pitch_deviation_value = 0.0014
longitudinal_variable_pitch_coefficient = 0.00038
total_pitch_coefficient = 0.00017
differential_total_pitch_coefficient = 0.00005
yaw_deviation_value = 0.0002
differential_total_pitch_yaw_coefficient = 0.00011
```

```
horizontal_tail_moment_coefficient_deviation = -0.0001
elevator_coefficient = -0.00001
vertical_tail_moment_coefficient_deviation = -0.0001
rudder_coefficient = -0.000005

propeller_data = {
    4: (200, 200),
    8: (1000, 700),
    12: (1900, 1100),
    16: (2400, 1400),
    20: (3200, 1900),
    24: (4000, 2500),
    28: (5800, 3700),
    32: (8200, 4500),
    36: (11000, 7000),
}

def get_propeller_force_torque(u_t):
    T, Q = propeller_data.get(u_t, (0, 0))
    return T, Q

def calculate_moments(u_c, u_cd, u_e, u_a, u_t, u_eh, u_av):
    C_rotor_roll = roll_deviation_value + lateral_pitch_roll_factor * u_a +
        differential_total_distance_roll_coefficient * u_cd
    M_rotor_roll = C_rotor_roll * 0.5 * rho * V_tip**2 * A * R
    C_vertical_tail_roll = vertical_tail_moment_coefficient_deviation +
        rudder_coefficient * u_av
    M_vertical_tail_roll = C_vertical_tail_roll * 0.5 * rho * V_tip**2 *
        S_vertical_tail * d_vertical_tail
    M_roll = M_rotor_roll + M_vertical_tail_roll

    C_rotor_pitch = pitch_deviation_value +
        longitudinal_variable_pitch_coefficient * u_e + total_pitch_coefficient
        * u_c + differential_total_pitch_coefficient * u_cd
    M_rotor_pitch = C_rotor_pitch * 0.5 * rho * V_tip**2 * A * R
    C_horizontal_tail_pitch = horizontal_tail_moment_coefficient_deviation +
        elevator_coefficient * u_eh
```

```
    M_horizontal_tail_pitch = C_horizontal_tail_pitch * 0.5 * rho * V_tip**2 *
        S_horizontal_tail * d_horizontal_tail

    T, Q = get_propeller_force_torque(u_t)
    M_propeller_pitch = T * d_propeller + Q
    M_pitch = M_rotor_pitch + M_propeller_pitch + M_horizontal_tail_pitch

    C_rotor_yaw = yaw_deviation_value +
        differential_total_pitch_yaw_coefficient * u_cd
    M_rotor_yaw = C_rotor_yaw * 0.5 * rho * V_tip**2 * A * R
    C_vertical_tail_yaw = vertical_tail_moment_coefficient_deviation +
        rudder_coefficient * u_av
    M_vertical_tail_yaw = C_vertical_tail_yaw * 0.5 * rho * V_tip
    M_vertical_tail_yaw = C_vertical_tail_yaw * 0.5 * rho * V_tip**2 *
        S_vertical_tail * L_vertical_tail
    M_yaw = M_rotor_yaw + M_vertical_tail_yaw
    return M_roll, M_pitch, M_yaw

def dynamics(y, t, I_roll, I_pitch, I_yaw, u_c, u_cd, u_e, u_a, u_t, u_eh,
    u_av):
    phi, theta, psi, phi_dot, theta_dot, psi_dot = y
    M_roll, M_pitch, M_yaw = calculate_moments(u_c, u_cd, u_e, u_a, u_t, u_eh,
        u_av)
    dphi_dot_dt = M_roll / I_roll
    dtheta_dot_dt = M_pitch / I_pitch
    dpsi_dot_dt = M_yaw / I_yaw
    return [phi_dot, theta_dot, psi_dot, dphi_dot_dt, dtheta_dot_dt,
        dpsi_dot_dt]

time_points = np.linspace(0, 20, 1000)

initial_conditions = [0, 0, 0, 0, 0, 0]

def objective(x):
    u_c, u_cd, u_e, u_a, u_t, u_eh, u_av = x
    dyn_func = lambda y, t: dynamics(y, t, I_roll, I_pitch, I_yaw, u_c, u_cd,
        u_e, u_a, u_t, u_eh, u_av)
    solution = odeint(dyn_func, initial_conditions, [0, 20])
    phi, theta, psi = solution[-1, :3]
```

```
    return abs(phi) + abs(theta) + abs(psi)

initial_guess = [0, 0, 0, 0, 0, 0, 0]
bounds = [(0, 30), (-25, 25), (-25, 25), (-25, 25), (0, 36), (-25, 25), (-25,
    25)]
result = minimize(objective, initial_guess, bounds=bounds, method='SLSQP')

u_c_opt, u_cd_opt, u_e_opt, u_a_opt, u_t_opt, u_eh_opt, u_av_opt = result.x
print("Optimal Control Inputs:")
print(f"u_c: {u_c_opt}, u_cd: {u_cd_opt}, u_e: {u_e_opt}, u_a: {u_a_opt}, u_t:
    {u_t_opt}, u_eh: {u_eh_opt}, u_av: {u_av_opt}")
```

Appendix D

Listing 6: The python source code for solving question four

```
from scipy.optimize import minimize
from scipy.integrate import odeint
import numpy as np

I_roll = 8000
I_pitch = 20000
I_yaw = 25000
rho = 0.9093
R = 6
A = np.pi * R**2
V_tip = 180
S_horizontal_tail = 1
d_horizontal_tail = -3
S_vertical_tail = 0.5
d_vertical_tail = -3
L_vertical_tail = 0.2
d_propeller = -0.2

propeller_data = {
    4: (200, 200),
    8: (1000, 700),
    12: (1900, 1100),
```

```
    16: (2400, 1400),
    20: (3200, 1900),
    24: (4000, 2500),
    28: (5800, 3700),
    32: (8200, 4500),
    36: (11000, 7000),
}

def get_propeller_force_torque(u_t):
    T, Q = propeller_data.get(u_t, (0, 0))
    return T, Q

moment_coefficients_mapping = {
    0: {
        'roll_deviation_value': 0.0002,
        'lateral_pitch_roll_factor': 0.00028,
        'differential_total_distance_roll_coefficient': -0.00005,
        'pitch_deviation_value': 0.001,
        'longitudinal_variable_pitch_coefficient': 0.0002,
        'total_pitch_coefficient': -0.00002,
        'differential_total_pitch_coefficient': 0.0001,
        'yaw_deviation_value': 0.0003,
        'differential_total_pitch_yaw_coefficient': 0.00009,
        'horizontal_tail_moment_coefficient_deviation': -0.0001,
        'elevator_coefficient': -0.000001,
        'vertical_tail_moment_coefficient_deviation': -0.00001,
        'rudder_coefficient': -0.000001
    },
    0.1: {
        'roll_deviation_value': 0.00035,
        'lateral_pitch_roll_factor': 0.00032,
        'differential_total_distance_roll_coefficient': -0.00019,
        'pitch_deviation_value': 0.0014,
        'longitudinal_variable_pitch_coefficient': 0.00038,
        'total_pitch_coefficient': 0.00017,
        'differential_total_pitch_coefficient': 0.00005,
        'yaw_deviation_value': 0.0002,
        'differential_total_pitch_yaw_coefficient': 0.00011,
        'horizontal_tail_moment_coefficient_deviation': -0.0001,
```

```
        'elevator_coefficient': -0.00001,
        'vertical_tail_moment_coefficient_deviation': -0.0001,
        'rudder_coefficient': -0.000005
    },
    0.2: {
        'roll_deviation_value': 0.0004,
        'lateral_pitch_roll_factor': 0.0004,
        'differential_total_distance_roll_coefficient': -0.00024,
        'pitch_deviation_value': 0.0029,
        'longitudinal_variable_pitch_coefficient': 0.00045,
        'total_pitch_coefficient': 0.00034,
        'differential_total_pitch_coefficient': -0.000001,
        'yaw_deviation_value': 0.0001,
        'differential_total_pitch_yaw_coefficient': 0.00008,
        'horizontal_tail_moment_coefficient_deviation': -0.0001,
        'elevator_coefficient': -0.00005,
        'vertical_tail_moment_coefficient_deviation': -0.00015,
        'rudder_coefficient': -0.000016
    },

    0.3: {
        'roll_deviation_value': 0.0004,
        'lateral_pitch_roll_factor': 0.00044,
        'differential_total_distance_roll_coefficient': -0.00025,
        'pitch_deviation_value': 0.0042,
        'longitudinal_variable_pitch_coefficient': 0.0005,
        'total_pitch_coefficient': 0.00043,
        'differential_total_pitch_coefficient': -0.000001,
        'yaw_deviation_value': 0.0001,
        'differential_total_pitch_yaw_coefficient': 0.00008,
        'horizontal_tail_moment_coefficient_deviation': -0.0001,
        'elevator_coefficient': -0.00013,
        'vertical_tail_moment_coefficient_deviation': -0.0002,
        'rudder_coefficient': -0.000028
    }
}
def get_nearest_coefficients(rar):
    nearest_rar = min(moment_coefficients_mapping.keys(), key=lambda k: abs(k
        - rar))
```

```
    return moment_coefficients_mapping[nearest_rar]

def calculate_moments(u_c, u_cd, u_e, u_a, u_t, u_eh, u_av, coefficients):
    coefficients = get_nearest_coefficients(rar)
    roll_deviation_value = coefficients['roll_deviation_value']
    lateral_pitch_roll_factor = coefficients['lateral_pitch_roll_factor']
    differential_total_distance_roll_coefficient =
        coefficients['differential_total_distance_roll_coefficient']
    pitch_deviation_value = coefficients['pitch_deviation_value']
    longitudinal_variable_pitch_coefficient =
        coefficients['longitudinal_variable_pitch_coefficient']
    total_pitch_coefficient = coefficients['total_pitch_coefficient']
    differential_total_pitch_coefficient =
        coefficients['differential_total_pitch_coefficient']
    yaw_deviation_value = coefficients['yaw_deviation_value']
    differential_total_pitch_yaw_coefficient =
        coefficients['differential_total_pitch_yaw_coefficient']
    horizontal_tail_moment_coefficient_deviation =
        coefficients['horizontal_tail_moment_coefficient_deviation']
    elevator_coefficient = coefficients['elevator_coefficient']
    vertical_tail_moment_coefficient_deviation =
        coefficients['vertical_tail_moment_coefficient_deviation']
    rudder_coefficient = coefficients['rudder_coefficient']

    C_rotor_roll = roll_deviation_value + lateral_pitch_roll_factor * u_a +
        differential_total_distance_roll_coefficient * u_cd
    M_rotor_roll = C_rotor_roll * 0.5 * rho * V_tip**2 * A * R
    C_vertical_tail_roll = vertical_tail_moment_coefficient_deviation +
        rudder_coefficient * u_av
    M_vertical_tail_roll = C_vertical_tail_roll * 0.5 * rho * V_tip**2 *
        S_vertical_tail * d_vertical_tail
    M_roll = M_rotor_roll + M_vertical_tail_roll

    C_rotor_pitch = pitch_deviation_value +
        longitudinal_variable_pitch_coefficient * u_e + total_pitch_coefficient
        * u_c + differential_total_pitch_coefficient * u_cd
    M_rotor_pitch = C_rotor_pitch * 0.5 * rho * V_tip**2 * A * R
    C_horizontal_tail_pitch = horizontal_tail_moment_coefficient_deviation +
        elevator_coefficient * u_eh
```

```
    M_horizontal_tail_pitch = C_horizontal_tail_pitch * 0.5 * rho * V_tip**2 *
        S_horizontal_tail * d_horizontal_tail

    T, Q = get_propeller_force_torque(u_t)
    M_propeller_pitch = T * d_propeller + Q
    M_pitch = M_rotor_pitch + M_propeller_pitch + M_horizontal_tail_pitch

    C_rotor_yaw = yaw_deviation_value +
        differential_total_pitch_yaw_coefficient * u_cd
    M_rotor_yaw = C_rotor_yaw * 0.5 * rho * V_tip**2 * A * R
    C_vertical_tail_yaw = vertical_tail_moment_coefficient_deviation +
        rudder_coefficient * u_av
    M_vertical_tail_yaw = C_vertical_tail_yaw * 0.5 * rho * V_tip
    M_vertical_tail_yaw = C_vertical_tail_yaw * 0.5 * rho * V_tip**2 *
        S_vertical_tail * L_vertical_tail
    M_yaw = M_rotor_yaw + M_vertical_tail_yaw
    return M_roll, M_pitch, M_yaw

def dynamics(y, t, I_roll, I_pitch, I_yaw, u_c, u_cd, u_e, u_a, u_t, u_eh,
    u_av, rar):
    phi, theta, psi, phi_dot, theta_dot, psi_dot = y
    M_roll, M_pitch, M_yaw = calculate_moments(u_c, u_cd, u_e, u_a, u_t, u_eh,
        u_av, rar)
    dphi_dot_dt = M_roll / I_roll
    dtheta_dot_dt = M_pitch / I_pitch
    dpsi_dot_dt = M_yaw / I_yaw
    return [phi_dot, theta_dot, psi_dot, dphi_dot_dt, dtheta_dot_dt,
        dpsi_dot_dt]

def objective(x, I_roll, I_pitch, I_yaw, initial_conditions, time_span, rar):
    u_c, u_cd, u_e, u_a, u_t, u_eh, u_av = x
    dyn_func = lambda y, t: dynamics(y, t, I_roll, I_pitch, I_yaw, u_c, u_cd,
        u_e, u_a, u_t, u_eh, u_av, rar)
    solution = odeint(dyn_func, initial_conditions, time_span)
    phi, theta, psi = solution[-1, :3]
    return abs(phi) + abs(theta) + abs(psi)

initial_guess = [0, 0, 0, 0, 0, 0, 0]
```

```
bounds = [(0, 30), (-25, 25), (-25, 25), (-25, 25), (0, 36), (-25, 25), (-25,
    25)]
total_time = 20
step = 0.5
time_points = np.linspace(0, total_time, int(total_time/step))
initial_conditions = [0, 0, 0, 0, 0, 0]
current_controls = initial_guess.copy()

for start_time in time_points:
    flight_speed = 80 + (100/20) * start_time
    rar = flight_speed / V_tip
    result = minimize(objective, current_controls, args=(I_roll, I_pitch,
        I_yaw, initial_conditions, [start_time, start_time + step], rar),
        bounds=bounds, method='SLSQP')
    current_controls = result.x
    print(f"Optimal Control Inputs at time {start_time} s:")
    print(f"u_c: {current_controls[0]}, u_cd: {current_controls[1]}, u_e:
        {current_controls[2]}, u_a: {current_controls[3]}, u_t:
        {current_controls[4]}, u_eh: {current_controls[5]}, u_av:
        {current_controls[6]}")
    dyn_func = lambda y, t: dynamics(y, t, I_roll, I_pitch, I_yaw,
        *current_controls, rar)
    solution = odeint(dyn_func, initial_conditions, [start_time, start_time +
        step])
initial_conditions = solution[-1]
```

Listing 7: The python source code for validating question four

```
initial_guess = [0, 0, 0, 0, 0, 0, 0]
bounds = [(0, 30), (-25, 25), (-25, 25), (-25, 25), (0, 36), (-25, 25), (-25,
    25)]
total_time = 20
step = 0.5
time_points = np.linspace(0, total_time, int(total_time/step) + 1)
control_params = []
attitude_angles = []
initial_conditions = [0, 0, 0, 0, 0, 0]
current_controls = initial_guess.copy()

for start_time in time_points[:-1]:
```

```
    flight_speed = 80 + (100/20) * start_time
    rar = flight_speed / V_tip
    result = minimize(objective, current_controls, args=(I_roll, I_pitch,
        I_yaw, initial_conditions, [start_time, start_time + step], rar),
        bounds=bounds, method='SLSQP')
    current_controls = result.x
    dyn_func = lambda y, t: dynamics(y, t, I_roll, I_pitch, I_yaw,
        *current_controls, rar)
    solution = odeint(dyn_func, initial_conditions, [start_time, start_time +
        step])
    initial_conditions = solution[-1]
    control_params.append(current_controls)
    attitude_angles.append(initial_conditions[:3])

control_params = np.array(control_params)
attitude_angles = np.array(attitude_angles)

plt.figure(figsize=(12, 6))
plt.plot(time_points[:-1], attitude_angles[:, 0], label='Roll Angle (phi)')
plt.plot(time_points[:-1], attitude_angles[:, 1], label='Pitch Angle (theta)')
plt.plot(time_points[:-1], attitude_angles[:, 2], label='Yaw Angle (psi)')
plt.xlabel('Time (s)')
plt.ylabel('Attitude Angles (radians)')
plt.title('Attitude Angles Over Time')
plt.legend()
plt.grid(True)
plt.show()

deviations = np.abs(attitude_angles)
average_deviations = np.mean(deviations, axis=0)
max_deviations = np.max(deviations, axis=0)
total_deviations = np.sum(deviations, axis=0)

print("Average Deviations (Roll, Pitch, Yaw):", average_deviations)
print("Maximum Deviations (Roll, Pitch, Yaw):", max_deviations)
print("Total Deviations (Roll, Pitch, Yaw):", total_deviations)
```

Listing 8: The python source code for validating question four

```
initial_guess = [0, 0, 0, 0, 0, 0, 0]
```

```
bounds = [(0, 30), (-25, 25), (-25, 25), (-25, 25), (0, 36), (-25, 25), (-25,
    25)]
total_time = 20
step = 0.5
time_points = np.linspace(0, total_time, int(total_time/step))
initial_conditions = [0, 0, 0, 0, 0, 0]
current_controls = initial_guess.copy()

for start_time in time_points:
    flight_speed = 80 + (100/20) * start_time
    rar = flight_speed / V_tip
    result = minimize(objective, current_controls, args=(I_roll, I_pitch,
        I_yaw, initial_conditions, [start_time, start_time + step], rar),
        bounds=bounds, method='SLSQP')
    current_controls = result.x
    print(f"Optimal Control Inputs at time {start_time} s:")
    print(f"u_c: {current_controls[0]}, u_cd: {current_controls[1]}, u_e:
        {current_controls[2]}, u_a: {current_controls[3]}, u_t:
        {current_controls[4]}, u_eh: {current_controls[5]}, u_av:
        {current_controls[6]}")

    u_c_history.append(current_controls[0])
    u_cd_history.append(current_controls[1])
    u_e_history.append(current_controls[2])
    u_a_history.append(current_controls[3])
    u_t_history.append(current_controls[4])
    u_eh_history.append(current_controls[5])
    u_av_history.append(current_controls[6])

    dyn_func = lambda y, t: dynamics(y, t, I_roll, I_pitch, I_yaw,
        *current_controls, rar)
    solution = odeint(dyn_func, initial_conditions, [start_time, start_time +
        step])
    initial_conditions = solution[-1]

plt.figure(figsize=(15, 10))

plt.subplot(3, 3, 1)
plt.plot(time_points, u_c_history, label='u_c')
```

```
plt.xlabel('Time (s)')
plt.ylabel('u_c')
plt.grid(True)
plt.title('Control Input u_c Over Time')

plt.subplot(3, 3, 2)
plt.plot(time_points, u_cd_history, label='u_cd')
plt.xlabel('Time (s)')
plt.ylabel('u_cd')
plt.grid(True)
plt.title('Control Input u_cd Over Time')

plt.subplot(3, 3, 3)
plt.plot(time_points, u_e_history, label='u_e')
plt.xlabel('Time (s)')
plt.ylabel('u_e')
plt.grid(True)
plt.title('Control Input u_e Over Time')

plt.subplot(3, 3, 4)
plt.plot(time_points, u_a_history, label='u_a')
plt.xlabel('Time (s)')
plt.ylabel('u_a')
plt.grid(True)
plt.title('Control Input u_a Over Time')

plt.subplot(3, 3, 5)
plt.plot(time_points, u_t_history, label='u_t')
plt.xlabel('Time (s)')
plt.ylabel('u_t')
plt.grid(True)
plt.title('Control Input u_t Over Time')

plt.subplot(3, 3, 6)
plt.plot(time_points, u_eh_history, label='u_eh')
plt.xlabel('Time (s)')
plt.ylabel('u_eh')
plt.grid(True)
plt.title('Control Input u_eh Over Time')
```

```
plt.subplot(3, 3, 7)
plt.plot(time_points, u_av_history, label='u_av')
plt.xlabel('Time (s)')
plt.ylabel('u_av')
plt.grid(True)
plt.title('Control Input u_av Over Time')

plt.tight_layout()
plt.show()
```

Appendix E

$Time(s)$	u_c	u_{cd}	u_e	u_a	u_t	u_{eh}	u_{av}
0.00	0.0541	-1.2507	-8.4499	-1.6207	0.0000	-0.0097	-0.0001154
0.51	0.0542	-1.2464	-8.4499	-1.6182	0.0000	-0.0097	-0.0001153
1.03	0.0541	-1.2535	-8.4499	-1.6223	0.0000	-0.0097	-0.0001154
1.54	0.0541	-1.2535	-8.4499	-1.6223	0.0000	-0.0097	-0.0001154
2.05	0.0541	-1.2320	-8.4499	-1.6101	0.0000	-0.0097	-0.0001152
2.56	0.0542	-1.2321	-8.4499	-1.6100	0.0000	-0.0097	-0.0001151
3.08	0.0543	-1.2322	-8.4497	-1.6104	0.0000	-0.0097	-0.0001152
3.59	0.0534	-1.4594	-8.4507	-1.7391	0.0000	-0.0097	-0.0001178
4.10	0.0550	-0.9271	-8.4489	-1.4370	0.0000	-0.0097	-0.0001117
4.62	0.0534	-1.5761	-8.4507	-1.8051	0.0000	-0.0097	-0.0001191
5.13	0.0558	-0.9107	-8.4479	-1.4288	0.0000	-0.0097	-0.0001110
5.64	0.0483	-1.5972	-8.4567	-1.8159	0.0000	-0.0097	-0.0001195
6.15	0.0650	-0.8963	-8.4373	-1.4186	0.0000	-0.0097	-0.0001117
6.67	0.0421	-1.6061	-8.4655	-1.8285	0.0000	-0.0097	-0.0001202
7.18	0.0695	-0.8931	-8.4337	-1.4096	0.0000	-0.0097	-0.0001078
7.69	0.0370	-1.5929	-8.4714	-1.8214	0.0000	-0.0097	-0.0001202
8.21	0.0739	-0.9332	-8.4285	-1.4172	0.0000	-0.0097	-0.0001098
8.72	0.0527	-1.5521	-8.4848	-1.8520	0.0000	-0.0098	-0.0001268
9.23	0.0878	-0.9484	-8.4439	-1.3681	0.0000	-0.0097	-0.0001041
9.74	0.0616	-1.5698	-8.4938	-1.8868	0.0000	-0.0098	-0.0001302
10.26	0.1012	-0.8919	-8.4478	-1.3301	0.0000	-0.0097	-0.0001022
10.77	0.0659	-1.6236	-8.5053	-1.9193	0.0000	-0.0098	-0.0001297
11.28	0.1212	-0.8925	-8.4623	-1.3310	0.0000	-0.0097	-0.0001017
11.80	0.0812	-1.5768	-8.5186	-1.8911	0.0000	-0.0098	-0.0001283
12.31	0.1263	-0.9346	-8.4662	-1.3573	0.0000	-0.0097	-0.0001013
12.82	0.0808	-0.8727	-8.5246	-1.4882	0.0000	-0.0098	-0.0001191
13.33	0.1281	-0.9515	-8.4697	-1.3694	0.0000	-0.0097	-0.0001019
13.85	0.1069	-4.9841	-8.5057	-3.8233	0.0000	-0.0098	-0.0001655
14.36	0.1509	4.5088	-8.4968	1.7330	0.0000	-0.0098	-0.0000182
14.87	0.1642	-6.9225	-8.4922	-4.8884	0.0000	-0.0098	-0.0001757
15.39	1.2204	4.3810	-9.8197	1.5516	0.0000	-0.0113	-0.0000172
15.90	1.6574	-6.8439	-9.3116	-4.7350	0.0000	-0.0107	-0.0001719
16.41	1.1851	4.2703	-9.9103	1.4522	0.0000	-0.0114	-0.0000846
16.92	1.6232	-6.7322	-9.4009	-4.6652	0.0000	-0.0108	-0.0001151
17.44	1.3328	4.5547	-9.7386	1.5971	0.0000	-0.0112	0.0001071
17.95	1.4492	-7.6976	-9.6033	-5.2385	0.0000	-0.0110	-0.0001173
18.46	1.4469	5.5200	-9.6059	2.2904	0.0000	-0.0110	0.0001136
18.97	1.3651	5.6017	-9.7010	2.1114	0.0000	-0.0112	0.0000894
19.49	1.5221	5.4280	-9.5185	2.3635	0.0000	-0.0109	0.0001266
20.00	1.3136	-25.0000	-9.7609	-15.3661	0.0000	-0.0112	-0.0006672