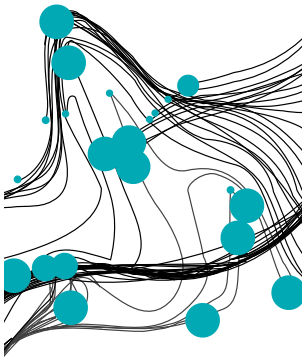
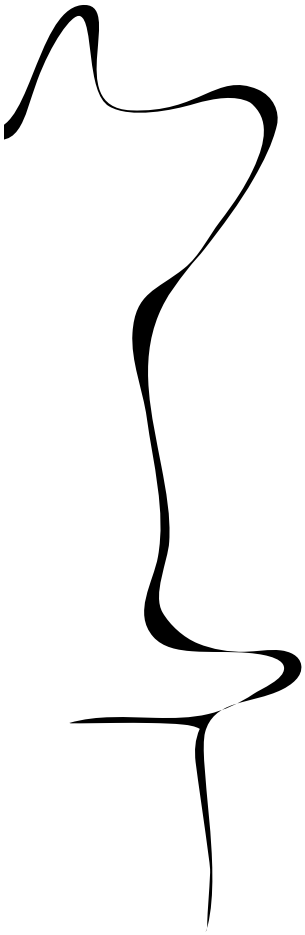


RAM ROBOTICS AND MECHATRONICS



SPINCOPTER: MODELING AND CONTROL

P.H.P.T.M. (Philippe) Damoiseaux

BSC ASSIGNMENT

Committee:
prof. dr. ir. A. Franchi
dr. S. Sun
Y. Shen
dr. ing. A. Lavrenko

July, 2023

030RaM2023
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Abstract

This paper explores the control strategy for a unique single rotor drone equipped with a fixed wing that incorporates a controllable control surface. It details the drone's modeling, the development of a Linear Quadratic Regulator (LQR) controller, and its implementation. The effectiveness of the control strategy is validated numerically, providing insights for future research in this field.

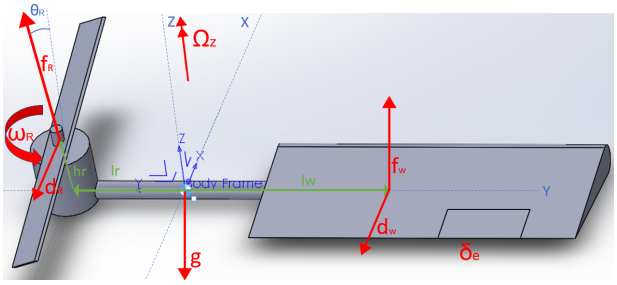


Fig. 1: System illustration of the Spincopter

1 Introduction

Drones with multiple rotors, like quadcopters or octocopters, suffer from a significant constraint due to their limited power reserves, which restricts their flight duration and operational distance. The momentum theory provides a theoretical framework for understanding this limitation, suggesting that a vehicle achieves greater aerodynamic efficiency when a larger volume of airflow is used for propulsion. The SpinCopter serves as a practical solution to this problem.

This drone tests the theory of sustaining its own weight through an upward force generated not only by its motor but also by a wing, which exerts the same effort as a motor propeller. It is built in a way so that it only consists of two controllable inputs, making it mechanically very simple. The two controllable inputs are the speed of the motor and the angle of the control surface. Figure 1 and 11 provide an overview of the drone. The figure shows a wing attached to a beam, with a motor fixed at its end, eventually creating a center of gravity between the motor and the wing.

This thesis is structured as follows: The first section provides an introduction and overview of the study. The second section describes the modeling process of the drone. The third section presents the control strategy. This is followed by a numerical analysis and a final section concluding the paper and suggesting areas for future research.

2 Modeling

The modeling process employs two types of coordinate systems: an inertial coordinate system known as E, which is fixed to the ground, and a body-fixed coordinate system is referred to as B [1]. These two frames will be differently noted by superscripts. For example the gravitational force $\boldsymbol{g}^E$ expresses $\boldsymbol{g}$ with respect to the inertial frame. For simplicity the superscript for the body frame are be omitted as most equations (unless specified) are in the body frame.

2.1 Forces on the drone

There are several forces acting on the drone. The main forces can be expressed in 4 equations. There are two components of the drone that create both drag and lift. For the purpose of this paper, the equations of these functions are simplified and given in equations 1,2, 3, 4.

$$f_w = c_l \Omega_z^2 \delta_e \tag{1}$$

$$d_w = c_d \Omega_z^2 \tag{2}$$

$$f_r = c_r \omega_R^2 \tag{3}$$

$$d_r = c_t \Omega_z \omega_R^2 \tag{4}$$

These 4 equations resemble the upwards force of the wing (f_w), drag of the wing (d_w), upwards force of the propeller (f_r) and drag of the propeller (d_r). All equations have their respective coefficient of lift and drag which is unique to the wing (c_l and c_d) and propeller (c_r and c_t). The motor generates a rotational force, ω_R, by rotating the propeller, which in turn initiates a rotational velocity of the drone, Ω_z. This rotational velocity affects all equations besides the equation initiating it. Furthermore, the lift of the wing can be controlled by altering the angle of

the control surface which is given in radians by δ_e. These two control inputs will allow the drone to hover and fly where desired. These equations, along with the equation of gravity, form the basis of this paper.

To properly model this, we examine both the dynamic and kinematic models. These are then further distinguished by identifying both the translational and rotational forces.

2.2 Translational

In the process of modeling, it's crucial to pinpoint the drone's location within the translational domain (x, y, z). To achieve this, we analyze the translational dynamics and kinematics using equations 5 and 6, derived from [2].

$$\dot{\boldsymbol{p}}^{\boldsymbol{E}} = \boldsymbol{R}\boldsymbol{v} \tag{5}$$

$$\dot{\boldsymbol{v}} = (\boldsymbol{\omega} \times \boldsymbol{v}) + \frac{1}{mass}\boldsymbol{F} - \boldsymbol{g} \tag{6}$$

Equation 6 allows us to calculate the drone's velocity and acceleration in all directions. The angular acceleration is computed by taking the cross product of the angular velocity and the translational velocity, and then adding the impact of the external forces. The cross product between the angular velocity and the velocity results in an acceleration. Dividing the force by the mass, according to Newton's second law, also yields an acceleration.

Once the angular acceleration is obtained in the body frame, it needs to be represented this in the inertial frame, as it is the drones operational frame. To transform the velocities, we employ the Rotation matrix (Z, X, Y), a standard rotation matrix in the field of drone dynamics. However, there are other options that can be chosen. This one is specifically chosen because it is conventional and the ease of conversion. The Rotation matrix is a matrix that, when multiplied with a vector, results in the rotation of the vector without altering its magnitude [3].

Within the world of modeling and control this matrix is used to convert forces acting on the body frame into the inertial frame. For this assignment we used the Z-X-Y Euler angles. This means that the conversion from body frame to inertial frame is in a certain order of turns/ operations. The inertial frame is first rotated about its z axis in the rotated frame through the yaw angle, ψ, which is followed by a rotation about its X axis in the rotated frame through the roll angle, ϕ, and a rotation about its X axis in the rotated frame through the pitch angle, θ [4].

2.3 Rotational

As the drone will be rotating constantly, the rotational dynamics and kinematics will be the most important to model. By considering the forces previously described, we can deduce their influence on rotational velocities and accelerations. The drone's rigid body equations of rotational motion [5] are represented in equations 7 and 8. We assume the body to be rigid, as it neither bends nor deforms in air.

$$\dot{\boldsymbol{R}}^{E} = \boldsymbol{R_T}\boldsymbol{\omega} \tag{7}$$

$$I\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times I\boldsymbol{\omega} + \boldsymbol{\tau} \tag{8}$$

Equation 8 shows the rotational velocity in the body frame for all axes. This formula is derived by identifying the velocity in a certain direction and subtracting it from the torques that are acting on the frame. Thereafter equation 7 represents the change in angles with respect to the inertial frame. This is done by multiplying it with the Euler Angle rates represented by $\boldsymbol{R_T}$.

The rate of change over time of the Euler angle vector equates to the vector of Euler angle speeds. The connection between these Euler angle speeds and the body's angular velocity is captured within the Euler angle rates matrix. When this matrix is multiplied by the vector of Euler angle speeds, it yields the angular velocity in the global coordinate system [3]. Thus, R is simply calculated using equation 9 and given in equation 10.

$$E_{ijk}(\phi, \theta, \psi) := [e^i, R_i(\phi)^T e^j, R_i(\phi)^T R_j(\theta)^T e^k] \tag{9}$$

$$\boldsymbol{R_T} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \tag{10}$$

The forces that contribute to the system's torque originate from both the rotor and the wing. Because

the motor is at an angle θ in between the x and z axis, the direct forces will only have an effect in the x and z directions. However, in the case of rotational dynamics, the torque is being considered which originates from the forces but has different properties due to its distance from the cg. Therefore, the forces in the z and x direction will result in a rotational force around the y axis. The system's torque in all axes is given in equations 11,12,13. These equations were derived by identifying all the forces in all directions and cross multiplying them with the distance from the center of gravity. This way the torque is represented accurately according to the model.

$$\tau_x = C_r l_r cos(\theta_R)\omega_R^2 - C_l \delta_e l_w r^2 \tag{11}$$

$$\begin{aligned}\tau_y = C_r\omega_R^2 h_r cos(\theta_R)sin(\theta_R) - h_r cos(\theta_R) \\ (C_r\omega_R^2 sin(\theta_R) + C_t\omega_R^2 r)\end{aligned} \tag{12}$$

$$\tau_z = l_r(C_r\omega_R^2 sin(\theta_R) + C_t\omega_R^2 r) - C_d l_w r^2 \tag{13}$$

To further ascertain the drone's angular velocity, a CAD model was constructed (appendix 7.3) to accurately determine a possible inertia matrix. This is determined to be the an identity matrix with the diagonals having different values. These values being $I_{xx} = 0.02$, $I_{yy} = 0.001$, $I_{zz} = 0.02$. Substituting all this information into equations 8 and 7, an effective model of the rotational dynamics and kinematics was derived.
The final state space equations for all states determined are given in Appendix 7.1.

2.4 Equilibrium

Having determined the equations for both the translational and rotational dynamics and kinematics, a hovering solution has to be determined. For simplicity of the assignment, a special equilibrium at which the drone will be hovering perfectly flat has been chosen to be pursued. In this case the desired values of all states besides the angular velocity r and the angle ψ should equal to 0. The drone will be constantly turning around its z axis as that is how the wing will be able to create lift and further approach the goal of the thesis which is to achieve a higher aerodynamic efficiency. In this case r will be a constant and the angle ψ is changing constantly as it depends on the velocity r.

When all states are 0 besides the ones mentioned above, the drone will be flying in a equilibrium which can also be identified as a hovering position where the position will remain constant. The parameters of this drone are therefore determined according to these requirements. When equaling the states and their respective integrals to 0 there are 6 state space equations which do not equal 0 without parameter optimization. These respective equations are for the states $\dot{p}, \dot{q}, \dot{r}, \dot{\psi}, \dot{u}\ \dot{v}$ and $\dot{w}$. These equations are given in section 7.2.

To further simplify things, we make hr 0 because else it would create problems for this specific hovering solution. by doing this we prevent a force in angular velocity around the y axis which would not be counteracted by other forces or would incur an angle change in other angles. This would have made the problem more complex.

2.5 Hovering solution

To identify the values of the parameters which will make this drone fly, initial guesses according to some quick online searches are made. These initial guesses with reasonable boundaries are plugged into a Matlab simulation which optimizes the values according to the functions equating to 0. The issue is that there are endless possibilities. This means that there are multiple local minima and Matlab usually focuses on one and further optimizes that one. Therefore, the simulations are very sensitive to initial guesses. To prevent this, a global function (Matlab's global search in combination with fmincon) is created which identifies multiple minima and further optimizes these to eventually find the most optimal one. The cost function analyzed with this script is given in equation 14.

$$cost = \sqrt{\dot{q}^2 + \dot{r}^2 + \dot{u}^2 + \dot{w}^2} \tag{14}$$

The final values obtained from the simulations are

given in table 1

ω_R	$596.20 rads/s^{-1}$
δ_e	$0.0916 rads$
θ_R	$0.1052 rads$
r_0	$10.5015 rads/s^{-1}$
C_l	0.6208
C_R	$1e^{-5}$
C_d	$1e^{-7}$
C_t	$-1e^{-7}$
lw	$0.1263m$
lr	$0.2243m$

Tab. 1: Final parameter list

3 Control strategy

To be sure that the drone can hover and be controlled into different directions, a controller has to be created and simulated. For this controller there are multiple strategies. A cascade controller is the optimal design since not all states of the drone are fully controllable. It also allows for the problem to be broken into two or three parts for ease of analysis [6]. This would be done through the use of an inner and outer loop which are both determined by an LQR function in Matlab.

The observability matrix of the drone does not need to be determined/calculated because all the states are fully observable.

3.1 Controllability

The drone's dynamics were initially modeled as a nonlinear system, which was then linearized around the equilibrium point (determined in section 2.5) corresponding to the hovering condition. This was done through calculating the jacobian of each equation with respect to their respective variable [7]. This gives a clear state space equation in the form of equation 15.

In this equation the variable "e" stands for error. It is the difference between the desired output and the actual output of the system. The goal of a control system is to minimize this error.

$\dot{e}$ represents the derivative of the error. Furthermore matrix A and B are constant coefficients of the states and coefficients that weight the inputs respectively [8]. The control input "u" is chosen to try to minimize the error "e" and its derivative $\dot{e}$.

$$\dot{\vec{e}} = A\vec{e} + B\vec{u} \tag{15}$$

The controllability of the drone's states was a crucial aspect of the control design process. Controllability refers to the ability to drive the system from any initial state to any desired final state in finite time, using the system inputs. The controllability of the system was analyzed by constructing a controllability matrix which is given in equation 16. This is a matrix that combines the effects of the system dynamics (A) and the control inputs (B) [9]. In this equation n represents the amount of states.

$$cc = [B, AB, A^2B, A^3B, ..., A^{n-1}B] \tag{16}$$

The controllability matrix was calculated using the state-space representation of the linearized drone dynamics. The rank of this matrix provides a measure of the number of controllable states [9]. In this case the matrix had rank 9 which means that there are 9 controllable states. The other states are not directly controllable through the 2 given inputs. However, this does not mean that they will not be able to be controlled at the end.

3.2 Full Controller

The stabilization of the drone during hovering flight was achieved by implementing a Linear Quadratic Regulator (LQR) controller (a control strategy that optimizes system performance by minimizing a quadratic cost function. It represents the deviation of system states and inputs from their desired values [10]). With the use of both the A and B matrix obtained in the previous paragraph, the LQR controller can be designed. The LQR design process involved defining a cost function that penalizes deviations from the desired state and control effort. The weights described in the Q matrix in the cost function were chosen to balance the trade-off between tracking performance and control effort.

Originally the linearized system was able to control 9 out of the 12 states namely the roll rate (p), yaw rate (r), roll angle (ϕ), pitch angle (θ), the body frame velocities (u, v, w) and the longitudinal positions (y,z). However, the remaining three states, namely the pitch rate (q), yaw angle (ψ) and longitudinal position (x), were not directly controllable through the two controllable inputs. To address this, natural damping in the pitch and roll rate are introduced and to control the longitudinal directions and a cascade controller is implemented to control the longitudinal state.

3.2.1 Cascade controller

To fully control the drone the full controller has to be a cascade controller which means that there is an inner and outer loop. However, for it to work properly, the inner loop in this case has to be the most sensitive to disturbances, work less fast than the outer loop and have less influence than the outer loop [11]. Because of these reasons, the inner loop for this drone will control the controllable angles and angular velocities while the outer loop will control the vector velocities. The angles and angular velocities are the most sensitive to disturbances and can be pushed out of equilibrium more easily than the longitudinal speeds. The composition of this controller can be seen in figure 2.

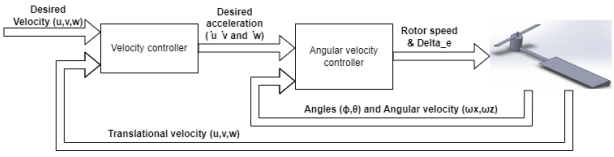


Fig. 2: Cascade Control diagram of Drone controller

To achieve this, the LQR controller gain of the inner loop is determined first before the outer loop gain. Considering that the inner loop is more sensitive to disturbances, the inner loop has to be stabilized before creating the outer loop. With the use of Matlab, a gain matrix (K) can be calculated for the inner loop. This is later applied to the outer loop.

3.2.2 LQR controller inner loop

The LQR controller in the inner loop controls the angles ϕ, θ and the angular velocities p and r. For this LQR controller a Q and R matrix are constructed to put priority on certain states. The Q matrix affects the controllable states and the R matrix affects the control inputs. Therefore we can determine some characteristics. The most important characteristics here were to ensure that the control surface δ_e does not exceed 0.7 or -0.7 radians (45°) as that is physically impossible and that the rotor speed does not change instantly or go negative as a motor can not spin that fast nor turn the in the other direction. Therefore the values in the R matrix are considerably bigger than the values in the Q matrix.

Furthermore, the rotation matrix has been determined by Z-X-Y and therefore the angle ϕ is the most influential. This means that the angle has to be brought to equilibrium as fast as possible. Lastly, the angular velocity around z (r) is also an important factor as that ensures the the drone creates lift and hovers. The values of both R an Q for the inner loop can be seen below.

$$\mathrm{Q} = \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 5000 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \quad \mathrm{R} = \begin{bmatrix} 90000 & 0 \\ 0 & 2000 \end{bmatrix}$$

3.2.3 LQR controller outer loop

The calculation of the outer loop gain matrix is done in the same way as the inner loop. However, for the outer loop, the A and B matrix are affected by and affect other states than that of the inner loop. In this case, the B matrix is constructed through the states controlled by the inner loop (p, r, ϕ and θ) and the A matrix is constructed through the states that still have to be controlled (u,v,w). Therefore the controllability has to be checked again and a new Q and R matrix have to be determined and evaluated.

For the Q matrix, three parameters are influenced. These parameters are u, v and w. From the formulas we can determine that the parameter w is the most sensitive compared to the other parameters. Further-

more the velocity v is very dependent on angles θ and ϕ. Because of this, the cost function also needs to be larger but because it is not influenced by the rotor it is less important than w.

With regards to the R matrix, the states being affected by it are the p, r, ϕ and θ. Because the states p and r are less sensitive, it does not need to be large compared to the cost of the other states. The angles need to be stabilized earlier than the angular velocities.

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 30 \end{bmatrix} \quad R = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

To properly control the location of the drone in the translational domain, it is possible to add another loop which controls the position the same way that the acceleration is controlled. Because only the altitude is not stable with the velocity loop, only a controller for the z domain has to be built. This is done through the following Q and R matrix. It still takes into account the x and y positions but those are not connected as can be seen in figure 3. The value of Q needs to be minimal else it will push everything out of equilibrium.

$$Q = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.001 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3.3 Simulink implementation

The cascaded control architecture was implemented using Simulink. The LQR controller was depicted as a gain block positioned ahead of the system, with a feedback loop supplying the control input. This feedback is subtracted from the mean and then fed into the gain matrix, transforming it into the suggested system input.

The outer loop controller was also portrayed as a gain block, with its output directed into the reference input of the LQR controller. The simulation results underscored the efficacy of the proposed control architecture in stabilizing the drone during hovering flight.

The complete architecture is illustrated in figure 3. This figure displays all three loops each followed by a different K gain component. These gain matrices generate an input for the rotor speed (ω_R) and control surface angle (δ_e), which are then fed into the non-linear system. The non-linear system outputs the states, which are subsequently integrated to derive the values for the speeds and angles.

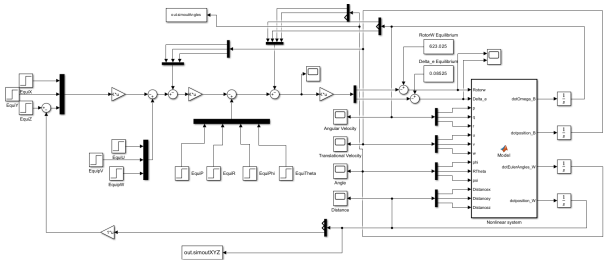


Fig. 3: Simulink controller setup

4 Numerical validation

Having obtained the full dynamic model and built a cascade LQR controller, the drone has to be properly simulated to observe the behaviour of the whole system. To analyse whether the system works accordingly, different tests are implemented. These tests consist out of testing the inner controller, the full controller (including all feedback loops) going to equilibrium and the full controller with minor disturbances. The goal with these tests is to see whether the drone will equalize and whether the drone will be able to recover from disturbances.

4.1 Control of innerloop

After establishing a gain matrix for the inner loop, simulations can only be run by setting the translational domain to zero. This is a temporary measure, as the full controller implementation will address this later. However, from the initial tests with only the inner controller, figures 4 and 5 demonstrate that the necessary states converge to equilibrium, with

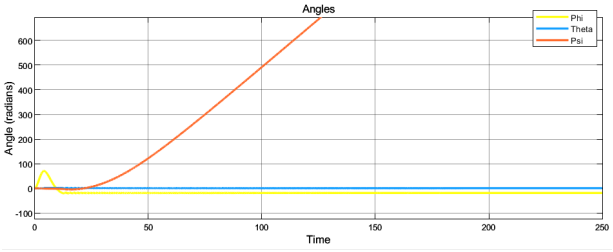


Fig. 4: Inner loop response of the angles

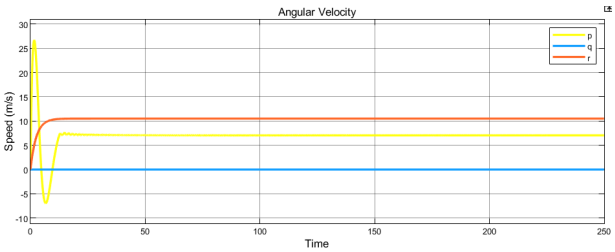


Fig. 5: Inner loop response of angular velocities

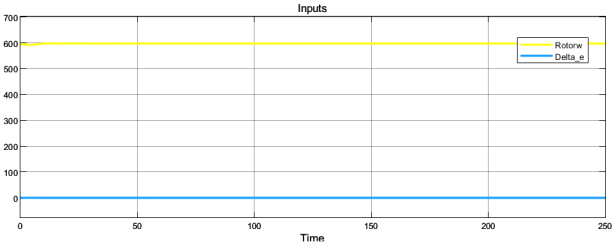


Fig. 6: Rotor speed (m/s) and δ_e (rad)

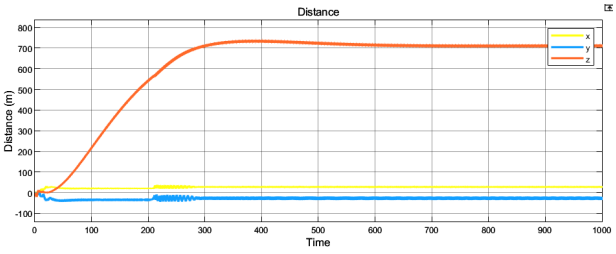


Fig. 7: X,Y and Z stabilization response

the yaw increasing at a steady rate. However, it is notable that the angular velocity p does not converge at 0 which is not what was expected. However, this has no effect on the final result as the angle ϕ is dependent on other parameters too and will still stabilize around a different value. These values are then implemented into the simulation so that it can converge more easily and reduce the error coming from the gain matrix.

Additionally, the results from figure 6 show that the inputs into the non-linear system are close to equilibrium and actively counteract any undesired activity. These three graphs demonstrate that, when not controlling the translational domain, the drone can hover and reach equilibrium. This shows that the controller built to control these states will force it back to equilibrium.

4.2 Control of full system

The implementation of the outer loops in the drone control system is an important aspect that enables precise position stabilization. Upon initiating the simulation, the drone does not instantaneously achieve stability. However, it starts from a state of rest and stabilizes over time, a process governed by the outer loops. These loops interact with the inner loop controls, thereby influencing the x, y, and z states of the drone.

The LQR system has been fine-tuned based on the results of the simulations, leading to the determination of the Q and R matrices, as detailed in section 3.2.3. These matrices play a role in modulating the input of the inner loop, thereby ensuring the x, y, and z states are stabilized. This is evident in figure 7, where it can be observed that the system eventually reaches a state of equilibrium.

One noteworthy observation is that the drone must first attain a certain altitude before the system stabilizes. This fails to be a problem simply because this can be pre-set in the drone's GPS, allowing it to operate at a normal altitude.

Upon examining figures 7 and 8, it is apparent that the values exhibit significant fluctuations. This is a consequence of the inner loop's sensitivity, which makes it challenging to control when the translational domain also requires stabilization. Despite these os-

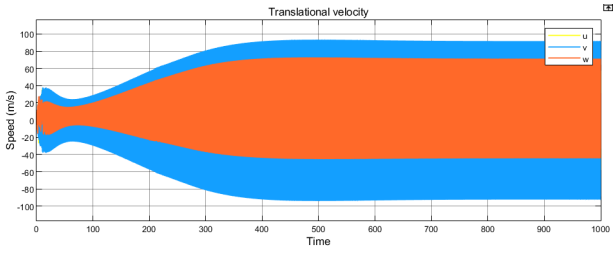


Fig. 8: X,Y and Z stabilization response

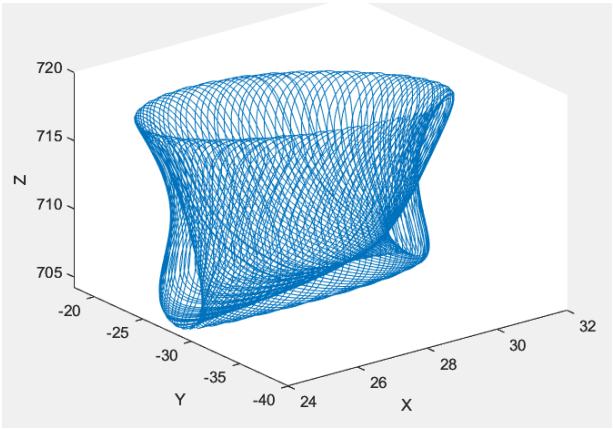


Fig. 9: displacement of the drone in x,y,z over time

cillations, the velocities do converge around zero, indicating that the system is stable around a specific point, despite this not being on the drone's body.

These oscillations and deviations from equilibrium enable the drone to maintain a unique equilibrium state, where it operates around a specific point. This behavior is depicted in figure 9. Over a duration of 100 seconds post-stabilization, the drone exhibits a pattern that is both self-sufficient and reliable. This pattern does not change over time showing that it is in a stable state. This is not a "real equilibrium" but could have been obtained because the model used to calculate the equilibrium is slightly different from the nonlinear simulation one. However, this is very normal in the real world because there will always be model mismatches.

4.3 Reaction to disturbances

A critical aspect of the drone control system is its ability to maintain stability under different kinds of disturbances. To evaluate this, we introduced various disturbances (sudden changes in x, y, z, ϕ, θ and ψ) into the simulation environment and observe the drone's response.

When a slight disturbance is brought to the x, y and z parameters the drone remains stable. However, the altitude (z) of the drone changes significantly but stabilizes at a new value. This means that the drone has multiple equilibrium's. Through further investigation and derivation it is possible to design a controller so that input controlling the altitude results in changes in multiple states but essentially controlling the z position.

5 Conclusion

To conclude, this paper presents the development and validation of a drone design with two control inputs, the SpinCopter, which leverages the principles of momentum theory to achieve greater aerodynamic efficiency. The drone's dynamics are modeled, and a cascade Linear Quadratic Regulator (LQR) controller is designed and implemented to stabilize the drone during hovering flight. The controller's performance is validated through numerical simulations, demonstrating the drone's ability to maintain stability under various disturbances. Despite the results suggesting that the SpinCopter holds promise for enhancing the capabilities of drone technology, it only withholds this in simulations. It is possible to determine from the data obtained in this paper that there is more research to be done into better modeling and control of the drone. However, this paper serves as a good start to any future research. Future work should focus on further refining the control strategy and exploring the drone's performance under a wider range of operational conditions.

6 Bibliography

References

[1] W. Zhang, M. W. Mueller, and R. D'Andrea, "A controllable flying vehicle with a single moving part," *IEEE International Conference on Robotics and Automation*, 2016.

[2] T. M. Randy Beard, *Small Unmanned Aircraft: Theory and Practice*.

[3] J. Diebel, "Representing attitude : Euler angles , unit quaternions , and rotation vectors," 2006.

[4] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[5] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 5, 2004, pp. 4393–4398 Vol.5.

[6] M. W. Müller and R. D'Andrea, "Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles," *The International Journal of Robotics Research*, vol. 35, pp. 873 – 889, 2016.

[7] A. D. Lewis and D. R. Tyner, "Geometric jacobian linearization and lqr theory," *The Journal of Geometric Mechanics*, vol. 2, pp. 397–440, 2011.

[8] D. Rowell, "2 . 14 analysis and design of feedback control systems state-space representation of lti systems," 2002.

[9] B. GOVIND, "Controllability and observability," 2016, unpublished lecture notes.

[10] S. Musa, "Techniques for quadcopter modelling design: A review," vol. 5, 05 2018.

[11] W. E. M. Company, "The benefits of cascade control," Tech. Rep.

7 Appendix

7.1 State space equations

$$\dot{p} = \frac{I_{yy}qr - I_{ZZ}qr - c_l\Delta_e * l_w r^2 C_r \omega_R L_r cos(\theta_R)}{I_{xx}} - c_p p \tag{17}$$

$$\dot{q} = -c_q q - \frac{r(C_t h_r cos(\theta_R)\omega_R^2 + I_{xx}p - I_{zz})}{I_{yy}} \tag{18}$$

$$\dot{r} = \frac{l_r(C_r\omega_R^2 sin(\theta_R) + C_t\omega_R^2 r) + I_{xx}pq - I_{yy}pq - C_d l_w r^2)}{I_{zz}} \tag{19}$$

$$\dot{\phi}^E = p - rsin(\theta) \tag{20}$$

$$\dot{\theta}^E = qcos(\phi) + rcos(\theta)sin(\phi) \tag{21}$$

$$\dot{\psi}^E = rcos(\theta)cos(\phi) - qsin(\phi) \tag{22}$$

$$\dot{u} = 9.81sin(\theta) - \frac{C_t * \omega_R^2 r + C_r sin(\theta_R)\omega_R^2 + C_d r^2}{mass} - qw + rv \tag{23}$$

$$\dot{v} = pw - ru - 9.81cos(\theta)sin(\phi) \tag{24}$$

$$\dot{w} = qu - pv + \frac{C_r cos(\theta_R) * \omega_R^2 + C_l\delta_e r^2}{mass} - 9.81cos(\theta)cos(\phi) \tag{25}$$

$$\dot{x}^E = w(sin(\phi)sin(\psi) + sin(\theta)cos(\phi)cos(\psi)) + ucos(\theta)cos(\psi) + vsin(\theta)cos(\psi)sin(\phi) \tag{26}$$

$$\dot{y}^E = v(cos(\theta)cos(\psi) + sin(\theta)sin(\phi)sin(\psi)) - w(sin(\theta)cos(\psi) - sin(\theta)cos(\phi)sin(\psi)) + ucos(\theta)sin(\psi) \tag{27}$$

$$\dot{z}^E = wcos(\theta)cos(\phi) - usin(\theta) + vcos(\theta)sin(\phi) \tag{28}$$

7.2 Equilibrium equations

$$\dot{q} = -\frac{-C_r l_r cos(\theta_R)\omega_R^2 + C_l \delta_e l_w r^2}{I_{xx}} \tag{29}$$

$$\dot{p} = \frac{C_t \omega_R^2 h_r r cos(\theta_R)}{I_{yy}} \tag{30}$$

$$\dot{r} = \frac{l_r(C_r \omega_R^2 sin(\theta_R) + C_t \omega_R^2 r) - C_d l_w r^2}{I_{zz}} \tag{31}$$

$$\dot{\psi} = r \tag{32}$$

$$\dot{u} = -\frac{C_t \omega_R^2 r + C_r sin(\theta_R)\omega_R^2 + C_d r^2}{mass} \tag{33}$$

$$\dot{w} = \frac{C_r cos(\theta_R)\omega_R^2 + C_l \delta_e r^2}{mass} - 9.81 \tag{34}$$

7.3 CAD model

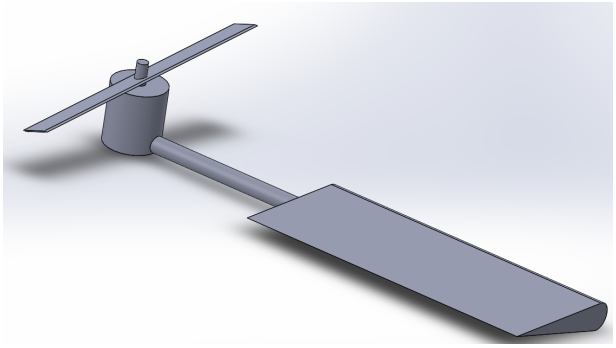


Fig. 10: Solidworks model

7.4 Model drawing

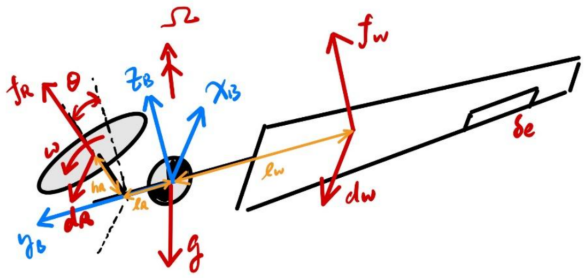


Fig. 11: Caption