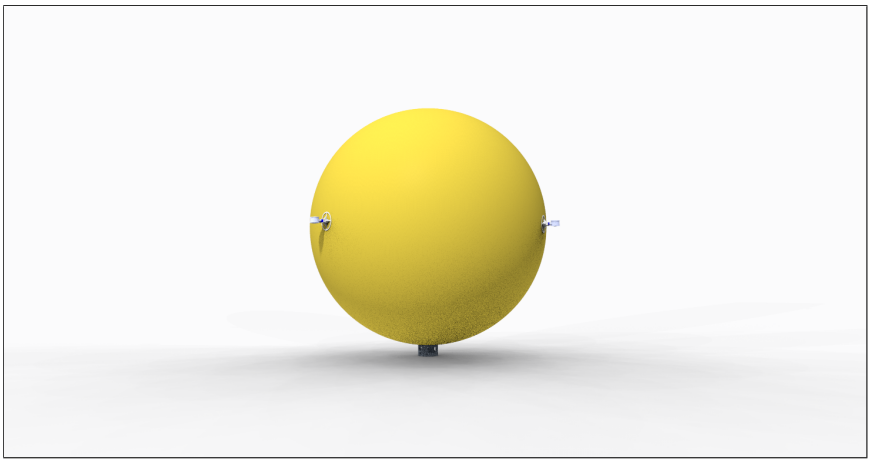
Imperial College London

AIM AIRSHIP

Autonomous Airship Design
Individual Report - GDP 2020



Student Name: Alexander Popov
CID: 01343523
Sub-team: FATT05 - Flight Dynamics and Control

Project Advisors: Dr. Rob Hewson
Prof. Sergei Chernyshenko
Dr. Siti Ros Shamsuddin
Dr. Daqing Yang
Dr. Yongyun Hwang
Dr. Rafael Palacios

FATT Supervisor: Dr. Daqing Yang
Department: Department of Aeronautics

Module: AERO96005 - GDP 2019-2020
Academic Year: 2019/2020

Date: June 15, 2020

Abstract

The nonlinear flight dynamics of an autonomous airship are derived and then Feedback Linearisation and Robust Feedback Linearisation are used to create two possible linear models of the system. A controller is then designed on the Feedback Linearised model using H_∞ Loop Shaping Design Procedure. The resulting control system is evaluated by running simulations for different possible flight paths. It is found that the airship's performance is greatly limited by the large buoyancy force from the envelope opposing the weight, and so a mass increase is proposed. It is concluded that any further improvements can be made by refining the control reference signals to design more favourable trajectories for the airship to follow.

CONTENTS

LIST OF FIGURES

I. Introduction

In any aerospace vehicle, there must be a control system which is capable of translating navigation commands into actuator inputs. The design of such a system relies on an understanding of the vehicle's physical dynamics, and how they are affected by different inputs.

In this case, a control system must be designed for a small autonomous airship. The design configuration is unconventional, and it is impossible to describe the dynamics without a substantial degree of uncertainty. This uncertainty places a fundamental limitation on the control design process, as the system can only be based around a model of the real dynamics. Ideally, the discrepancies between the model and the real airship system will not affect the control performance, however this is only possible in theory. To suppress any adverse effects, a controller should be designed to optimise for robustness in the closed-loop system, whilst still being able to accurately implement the desired navigation.

This report details the construction of the nonlinear dynamics equation, and the subsequent generation of two powerful linearised models. These models are then used to design a robust controller using H_∞ methods. To comprehensively assess the controller, it is then implemented in a simulation of the airship system, and the vehicle's ability to follow different paths and manoeuvres is analysed.

II. Flight Dynamics

A. Initial Configuration

The initial design configuration was a conventional ellipsoid airship, to be controlled either with control surfaces, such as a rudder and an elevator, or with thrusting apparatus. The dynamics of such a vehicle have several similarities with that of an aircraft, however there are a few key differences [1]. Firstly, the lift is generated through a buoyancy force, which results from the use of a low density gas (in this case, helium) within the airship envelope. Secondly, the lack of wings combined with several differences in shape result in different coupling behaviour to that of an aircraft - perhaps the most obvious being that roll and yaw are completely independent of each other. And lastly, the relatively low mass of an airship means that the displacement of air as it accelerates or decelerates through the environment will have a significant effect on the vehicle dynamics. The result is a phenomenon known as added mass, which will be explained in greater detail later.

B. Amended Configuration

It was quickly realised that an ellipsoid design may not be suitable for the mission requirements, and so the configuration was updated to a spherical envelope, with three tilting thrusters placed 120° apart at the equator, and a gondola below. To construct a model of the flight dynamics, the system could now be treated like a tricopter, combined with the buoyancy force, drag and added mass effect of an airship. The derivation of the tricopter dynamics uses the methods described in [2] and [3].

C. Defining the Problem

The forces acting on the airship are shown in Figure 1. Drag is not included in the diagram, as it does not act in a fixed direction, but its contribution is detailed later. The centre of buoyancy is assumed to be at the centre of volume of the envelope, with the centre of mass at some vertical distance below. Each propeller generates a thrust, which is assumed to take the form $f_i = k_f \omega_i^2$, where the subscript i denotes the i'th propeller, ω_i is the corresponding rotational velocity of the blades, and k_f is a constant design property. A torque is also produced around each motor, given by $t_i = k_t \omega_i^2$, where k_t is another design constant. Each motor has an associated tilt angle α_i, shown in the figure in a positive sense.

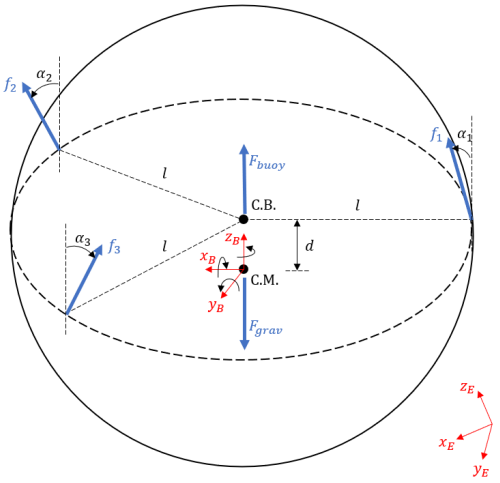


Fig. 1: The forces on the airship, excluding drag

This model makes use of two sets of axes, corresponding to two different reference frames. First we define body axes, with an origin at the centre of mass. This system describes a body-fixed frame, and the positive direction of each axis is shown in Figure 1. Additionally we set earth axes, which describe an inertial frame. For the purpose of our dynamics model the origin can be at any arbitrary fixed point in space, provided that the relative position and orientation of the body axes can be defined. The two coordinate systems are constructed in such a way that we can transform from one to the other using Euler rotation angles, such that the orientation of the body axes can be expressed in terms of a roll (ϕ), pitch (θ) and yaw (ψ) with respect to the earth axes. The positive directions of these angles are also shown in Figure 1. In order to preserve the conventional Euler relations whilst also setting the z-axis to be positive in the upwards direction (which is more convenient with the airship's thruster arrangement) the sign conventions typical of aircraft dynamics models cannot be replicated here. Now we can define the matrix Q, which maps the body axes to the inertial frame:

$$Q = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \quad (1)$$

where $c\phi$ and $s\phi$ represent $\cos\phi$ and $\sin\phi$ respectively. A useful property of this matrix is that $Q^{-1} = Q^T$.

D. Forces

For reasons that will be made more clear later, all forces are expressed in the body-fixed reference frame. An expression for the overall thrust vector could be formed by considering the geometry of the problem and resolving into components. It is shown in (2):

$$\boldsymbol{F}_{prop} = k_f H_f \boldsymbol{\rho}, \quad (2)$$

where

$$H_f = \begin{bmatrix} 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ -1 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{3}$$

and

$$\boldsymbol{\rho} = \begin{bmatrix} \omega_1^2 \sin\alpha_1 \\ \omega_2^2 \sin\alpha_2 \\ \omega_3^2 \sin\alpha_3 \\ \omega_1^2 \cos\alpha_1 \\ \omega_2^2 \cos\alpha_2 \\ \omega_3^2 \cos\alpha_3 \end{bmatrix} \tag{4}$$

The forces due to buoyancy and gravity can be expressed very conveniently in the inertial frame, and these can be transformed into body-fixed vectors by applying Euler rotations. These are shown respectively in (5a) and (5b), where b is the magnitude of the buoyancy force, m is the mass of the airship and g is the gravitational constant.

$$\boldsymbol{F}_{buoy} = Q^T \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix}, \quad \boldsymbol{F}_{grav} = Q^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{5a,b}$$

Finally we consider a drag force on the body. Although this is expected to be small due to the low operating speed (it is designed not to exceed $1.5\mathrm{m/s}$), the buoyancy and weight almost balance each other perfectly, so the thrust requirements are quite low. This suggests that even a modest drag force could still have a significant effect in certain cases. In this model, drag is always assumed to exactly oppose the direction of travel,and the symmetry of the design allows us to take the C_D estimate from the aerodynamics group as a constant value in any direction. Hence the drag force can be modelled as:

$$\boldsymbol{F}_{drag} = -\frac{1}{2}\rho_{air} C_D \begin{bmatrix} u \cdot |u| \\ v \cdot |v| \\ w \cdot |w| \end{bmatrix} \tag{6}$$

Now the total force on the system can be written as:

$$\boldsymbol{F} = \boldsymbol{F}_{prop} + \boldsymbol{F}_{buoy} + \boldsymbol{F}_{grav} + \boldsymbol{F}_{drag} \tag{7}$$

E. Moments

Just as with the forces, all moments are defined in the body-fixed frame. As the buoyancy and gravity forces lie on the same line of action, and the distance between the centre of mass and the drag line is assumed to be small, the total moment vector around the centre of mass is treated as being purely due to the propulsion systems. The most obvious contribution to this is due to the thrust forces acting at a distance from the centre. This component, $\boldsymbol{M}_{prop}$, can be found by combining the appropriate distance and force vectors to give the expression in (8), using the same structure as the thrust in (2):

$$\boldsymbol{M}_{prop} = k_f H_f \boldsymbol{\rho}, \tag{8}$$

where

$$H_t = \begin{bmatrix} d & -\frac{1}{2}d & -\frac{1}{2}d & 0 & -\frac{\sqrt{3}}{2}l & \frac{\sqrt{3}}{2}l \\ 0 & \frac{\sqrt{3}}{2}d & -\frac{\sqrt{3}}{2}d & l & -\frac{1}{2}l & -\frac{1}{2}l \\ l & l & l & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

and $\boldsymbol{\rho}$ is defined in (4).

Additionally, as the propeller blades rotate there is an associated torque in the direction of propeller rotation about the thrust axis. Thus in the body-fixed frame, the torque acting on the airship from each motor is acting opposite to the direction of propeller motion, and can be attributed to aerodynamic drag over the blades as they spin. In this design, all three propellers rotate clockwise, so each of the three drag torque components acts in the anticlockwise direction, which is defined as positive for this system. Since the torque vectors are in the same direction as the thrust vectors, the total drag torque contribution can be expressed as:

$$\boldsymbol{M}_{drag} = k_t H_f \boldsymbol{\rho} \tag{10}$$

The total moment about the body's centre of mass can now be determined:

$$\boldsymbol{M} = \boldsymbol{M}_{prop} + \boldsymbol{M}_{drag} \tag{11}$$

F. Equations of Motion

Having found the forces and moments acting on the airship, these can then be applied to the famous relations of Newton and Euler respectively to express linear and rotational accelerations. First we consider Newton's equation, applied to a system with velocity $\boldsymbol{\nu}$, angular velocity $\boldsymbol{\omega}$, mass M and force vector $\boldsymbol{F}$ as defined in (7):

$$\boldsymbol{F} = M(\dot{\boldsymbol{\nu}} + \boldsymbol{\omega} \times \boldsymbol{\nu}) \tag{12}$$

In general the value of M is simply taken as equal to the mass of the body (which we denote m), however in reality additional force is also required to move the air in the body's path. Therefore it would be more accurate to also account for the mass of the air that is displaced by the volume of the body as it moves. When describing a conventional aircraft, this term can be neglected, as it will be very small compared to the mass of the aircraft itself. However an airship is designed to be very light, such that the mass of displaced air may be very similar to that of the body. Therefore it is necessary to account for this by introducing an 'added mass' term, as described in [1]. By having to displace a relatively significant mass of air, the airship acts as if it is heavier that it its nominal mass when accelerating. To be precise, the added mass effect is described as a series of acceleration derivatives of the form $\frac{\partial F_k}{\partial a_k}$, where F_k is some force component in direction k, and a_j is an acceleration term, where j can define any of the three orthogonal directions (including k). Since the main body of the airship is spherical so fully symmetric, many of these acceleration derivatives reduce to zero, leaving only the three derivatives for which $j = k$, for example $\frac{\partial F_x}{\partial \dot{u}}$. The symmetry of the problem means that these three terms will be equal to each other, so they can all be set as a single term m_{added}, which is defined as:

$$m_{added} = \frac{2}{3}\pi \rho_{air} l^3 \tag{13}$$

Hence we can now amend (12) to give:

$$\boldsymbol{F} = (m + m_{added})\dot{\boldsymbol{\nu}} + m\boldsymbol{\omega} \times \boldsymbol{\nu} \tag{14}$$

Rotational motion is described with Euler's equation, where I is the inertia matrix of the airship with respect to the body-fixed axes:

$$\boldsymbol{M} = I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} \tag{15}$$

Symmetry dictates that all the rotational acceleration derivatives reduce to zero, so (15) can be left unchanged. (14) and (15) can be rearranged to give $\dot{\boldsymbol{\nu}}$ and $\dot{\boldsymbol{\omega}}$ as functions of $\boldsymbol{\nu}$, $\boldsymbol{\omega}$, the actuator input vector $\boldsymbol{\rho}$, the airship orientation and fixed design parameters. To construct a state space model, a system of the form shown in

(16) is required, where $\boldsymbol{x}$ is the vector of states, $\boldsymbol{u}$ is the input vector and $\boldsymbol{f}$ represents the system of nonlinear equations.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{16}$$

Using the rearrangements outlined above, it is clear that the roll, pitch and yaw rates $\boldsymbol{\omega} = \begin{pmatrix} p & q & r \end{pmatrix}^T$ can be defined as states, as can the velocities $\boldsymbol{\nu} = \begin{bmatrix} u & v & w \end{bmatrix}^T$. In keeping with (14) and (15), these are set in the body-fixed frame.

Additionally, the dependence on airship orientation (through the use of Euler's rotation matrix Q) means that the roll, pitch and yaw angles, $\boldsymbol{\eta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ should also be included as states. By definition, these are expressed in the inertial frame. The associated time-derivatives can be expressed as $\dot{\boldsymbol{\eta}} = K\boldsymbol{\omega}$, where K is a matrix in terms of ϕ and θ, and can be derived by considering the time-derivative of an Euler angle transformation. Although not shown here, K is presented explicitly in numerous publications, such as [3]. Finally, the position vector $\boldsymbol{\lambda} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is added to the state space system. As this is also relative to the earth axes, the derivative is simply $\dot{\boldsymbol{\lambda}} = Q\boldsymbol{\nu}$.

Considering the equations of motion and the transformation equations detailed above, the full dynamics model of the system can now be presented:

$$\dot{\boldsymbol{\nu}} = \frac{1}{m + m_{added}} \boldsymbol{F} - m\boldsymbol{\omega} \times \boldsymbol{\nu} \tag{17}$$

$$\dot{\boldsymbol{\omega}} = I^{-1}(\boldsymbol{M} - \boldsymbol{\omega} \times I\boldsymbol{\omega} \tag{18}$$

$$\dot{\boldsymbol{\eta}} = K\boldsymbol{\omega} \tag{19}$$

$$\dot{\boldsymbol{\lambda}} = Q\boldsymbol{\nu} \tag{20}$$

Hence we can write the state vector as:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \\ \boldsymbol{\eta} \\ \boldsymbol{\lambda} \end{bmatrix} \tag{21}$$

Furthermore, the system's input vector is set as $\boldsymbol{u} = \boldsymbol{\rho}$, as defined in (4). Not only can any desired motion of the airship be described with a combination of these states, their time-derivatives and the inputs, but also it is feasible that all twelve state values can be tracked in real time with sensors - this will be advantageous when designing a fully autonomous control system.

G. Uncertainties in the Nonlinear Model

The analysis outlined above provides a reasonably good model of the airship's flight dynamics, however there is no description of the interaction between the tricopter-like system and the helium-filled envelope. The presence of the envelope is accounted for with the buoyancy and drag terms, but these are treated as point loads. Clearly the envelope will have some damping effect on the vehicle's behaviour in roll, pitch and yaw due to skin friction drag. Ideally this damping would be quantified with a series of stability derivatives [1], however no accurate method for estimating these values could be found. This uncertainty can be reduced in subsequent design iterations by carrying out wind tunnel tests on a prototype. This would improve the reliability of the dynamics model, hence providing a stronger foundation on which to construct the control system.

III. The Linearised Model

A. Motivation

Having constructed a model of the airship's dynamics, this can be used to design a control system. The general form of a closed-loop control system is shown in Figure 2, where:

- u is the control input - this is the result of the system's control law, defining the required inputs to control the system;
- w is the exogenous input - this includes any external signals which influence the system, such as reference signals,load disturbances and sensor noise;
- y is the measured output - this consists of any states which are being tracked (for example, by sensors). It determines the actions required by the controller;
- z is the output to be controlled - this can be the tracking errors between states and their desired (reference) values, or it can be actuator signals

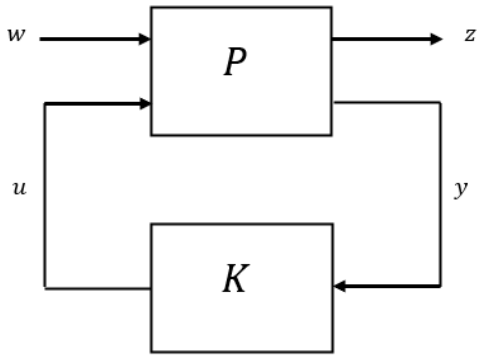


Fig. 2: The general structure of a control system

The system being controlled, P, is known as the plant. This represents the dynamics of the real system, but, for the purpose of control design and simulation, a dynamics model is used. The plant is controlled by a controller, K, which is a matrix of transfer functions which can determine the required control input based on y. For relatively complex systems such as the airship, the functions in K have to be chosen with large algorithms, with the plant as an input. However, these systematic methods can only work with a linear model. The system derived in the previous section is nonlinear, meaning that it has the general form:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}), \end{aligned} \tag{22}$$

where $\boldsymbol{f}$, $g(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ are functions of the states, and $\boldsymbol{y}$ is the same output vector as defined above.

To linearise this system, it must be manipulated into the form:

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} + D\boldsymbol{u} \end{aligned} \tag{23}$$

where A, B, C and D are matrices in terms of design parameters and physical constants.

B. Jacobian Linearisation

The most conventional linearisation technique involves defining A, B, C and D as a set of Jacobian matrices evaluated at an

equilibrium point in the state space. The concept is analogous to linearisation of scalar functions with Taylor series expansions. In this case, the highly nonlinear nature of the system, combined with the extensive coupling between states, exposed the limitations of the Jacobian method. Above all, such a linearisation requires selection of an appropriate equilibrium point, and the resulting model is only valid for small perturbations around that point. Defining such constrained operational regions for the airship would give an oversimplified model, unable to account for the large perturbations that could occur due to asymmetry of the propulsion configuration combined with the intricate coupling behaviour.

C. Feedback Linearisation

To overcome the limitations of the Jacobian approach, a much more versatile technique known as Input/Output Feedback Linearisation (FBL) was employed [4]. This method produces a 'virtual' linear system via transformations to the states and inputs. The controller is then designed for this virtual system, and the new control input can then be transformed back into a signal associated with the real system. This is done by integrating a 'linearisation control law' into the system architecture, as shown in Figure 3. FBL has several advantages over the more traditional techniques. Unlike Jacobian linearisation, the use of transformations on the nonlinear system means that no approximations need to be made to form the new model, and hence under certain conditions an exact linearisation is possible. Furthermore, the method does not rely on any knowledge of physical equilibrium conditions, and the resulting linearised system is valid at any operating point. An additional advantage of FBL is that the virtual linear system is decoupled, so each input only corresponds to one output. Strictly speaking, this is not a necessity for good control design, but it makes it easier to analyse the system responses (for example, to a step input) as the plots are fully independent of each other. This allows a designer to improve the closed-loop behaviour for certain states without having to consider adverse consequences elsewhere in the system.

D. Implementation

To implement Input/Output Feedback Linearisation, the consecutive time-derivatives of the measured output $\boldsymbol{y}$ are found until a derivative can be written in such a way that the control input $\boldsymbol{u}$ appears explicitly in the expression. The number of times $\boldsymbol{y}$ must be differentiated for this condition to be met is known as the relative degree of the system, denoted as r. For a multiple input, multiple output (MIMO) system such as the airship ($\boldsymbol{u}$ is a vector of six elements, and, to reduce the complexity of the model, the output $\boldsymbol{y}$ should be the same size), the total value of r is the sum of each degree r_i associated with the i'th output ($y_i = h_i(\boldsymbol{x})$). If $r = n$, where n is the number of states in the model, the system can be linearised exactly - it is not too difficult to show that the airship satisfies this condition when the output is set as $\boldsymbol{y} = \begin{bmatrix}\boldsymbol{\eta}^T & \boldsymbol{\lambda}^T\end{bmatrix}^T$ [3]. For each of the six outputs, $r_i = 2$, meaning that $r = 12 = n$. This is shown in (24), where $\boldsymbol{u}$ appears explicitly.

$$\ddot{\boldsymbol{y}} = \boldsymbol{L} + M\boldsymbol{u}, \tag{24}$$

where

$$\boldsymbol{L} = \begin{bmatrix} (\dot{K} - KI^{-1}\boldsymbol{\omega} \times I)\boldsymbol{\omega} \\ \frac{1}{m+m_{added}} Q(\boldsymbol{F}_{grav} + \boldsymbol{F}_{buoy} + \boldsymbol{F}_{drag}) \end{bmatrix} \tag{25}$$

and

$$M = \begin{bmatrix} KI^{-1}(k_f H_t + k_t H_f) \\ \frac{k_f}{m+m_{added}} Q H_f \end{bmatrix} \tag{26}$$

The condition $r = n$ implies exact linearisation because, by setting the transformed state vector $\boldsymbol{\xi}$ as consecutive derivatives of $h_i(\boldsymbol{x})$ up to the $(r_i - 1)$'th derivative, and by setting the transformed input as the r_i'th derivative of $h_i(\boldsymbol{x})$, all the nonlinearities associated with the state vector are cancelled out, and the only nonlinearity is in the transformation equation between $\boldsymbol{u}$ and $\boldsymbol{v}$, which is the linearisation control law. Applying these ideas to this example, we define $\boldsymbol{\xi}$ and $\boldsymbol{v}$ as follows:

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\eta} \\ \dot{\boldsymbol{\eta}} \\ \boldsymbol{\lambda} \\ \dot{\boldsymbol{\lambda}} \end{bmatrix}, \quad \boldsymbol{v} = \ddot{\boldsymbol{y}} = \begin{bmatrix} \ddot{\boldsymbol{\eta}} \\ \ddot{\boldsymbol{\lambda}} \end{bmatrix} \tag{27a,b}$$

The transformed, virtual system is linear:

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \begin{bmatrix} 0 & I_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{\xi} + \begin{bmatrix} 0 & 0 \\ I_3 & 0 \\ 0 & 0 \\ 0 & I_3 \end{bmatrix} \boldsymbol{v} \\ \boldsymbol{y} &= \begin{bmatrix} I_3 & 0 & 0 & 0 \\ 0 & 0 & I_3 & 0 \end{bmatrix} \boldsymbol{\xi} \end{aligned} \tag{28}$$

To structure of (28) can be compared to (23). In this case, the term D can be written as a matrix of zeros. This indicates that the controller action depends on current states, but is not affected by current control inputs.

Most importantly, by rearranging (24) and substituting $\ddot{\boldsymbol{y}} = \boldsymbol{v}$, the linearisation control law can be expressed:

$$\boldsymbol{u} = M^{-1}(\boldsymbol{v} - \boldsymbol{L}) \tag{29}$$

E. Robust Feedback Linearisation

Despite its many advantages, the classical Feedback Linearisation method is not very flexible to uncertainties in the physical dynamics model. This means that any substantial discrepancies between the the model and the real system will restrict the ability to design an effective control system. Uncertainties due to drag effects have already been discussed - without reliable wind tunnel data, the use of FBL may be limited. Furthermore, the virtual transformed system has little physical meaning, so when constructing a controller, it is not necessarily obvious how to relate changes in the control design to physical properties of the real system.

These issues can be avoided by using a more advanced version of the above method known as Robust Feedback Linearisation (RFBL) [5]. This follows the same general concept as classical FBL, but uses an extended linearisation law and state transformation by considering a system with all states at the origin ($\boldsymbol{x} = \boldsymbol{0}$). By fixing the transformation at a given physical point, the resulting virtual system is not so abstract in definition. Despite now linearising around a given location in state space, the resulting model is still valid for any operational point, regardless of distance from the origin. Furthermore, this method gives a more robust model, meaning that it should be more flexible to uncertainties in the physics. A complete expression of the linearisation law is given in [6].

For the subsequent control design procedure, both linearised models were used - the FBL model is less robust, but the RFBL model is limited by its greater computational complexity.

IV. Control System Design

A. System Architecture

The primary purpose of the control system is to ensure that the airship follows the desired path, as defined by a navigation algorithm, by adjusting the propeller thrusts and motor tilt angles accordingly. The structure of the system is shown in Figure 3. The desired path will be defined as a series of reference signals, indicating the required position and orientation of the vehicle at a given time. These are compared to the six current values, which can be estimated from the sensors. By subtracting these values from the desired values, the current tracking error is found. The controller then attempts to reduce this tracking error to zero, as this indicates that the airship is matching the desired trajectory. This control law gives the required control input into the virtual linear model. The linearisation control law transforms this into the required input for the real system, which can then be converted into a series of motor voltages.

B. The Simulated System

For the purpose of the design process, the system dynamics must be simulated using the nonlinear model, replacing sensor measurements with estimated outputs. The controller is designed according to this simulated system, shown in Figure 4, but it is implemented on board the airship with the real system shown in Figure 3.

C. Design Objectives

As well as good tracking of a desired trajectory, the design of a controller can be evaluated by the following criteria:

- The controller should be able to match the desired path in an efficient route that does not require aggressive or unnecessary work from the actuators;
- The tracking error performance should not worsen for more complex reference signals, such as turns and sudden stops;
- The controller should be robust, meaning that it is as valid in the real system as it is in the linearised model, despite the uncertainties in the latter;
- The controller should attenuate load disturbances (e.g. wind) and sensor noise to preserve reliable performance;
- The airship should stay as level as possible to avoid added uncertainties in sensor readings;
- The controller should not be so complex that the required processing power is too large to be implemented on the Raspberry Pi.

For this system, four controller design methods were considered: Model Predictive Control (MPC), Linear Quadratic Regulator (LQR), H_∞ Loop Shaping Design Procedure (LSDP), and Neural Predictive Control (NPC). This report will discuss the H_∞ design.

D. H_∞ Loop Shaping Design Procedure (LSDP)

H_∞ control methods find a controller K which yields a stable closed-loop system, ensuring that control input responses will converge to a steady state [7]. The controller is then optimised to give the largest possible robust stability margin ϵ_{max} - this quantity measures the ability of the controller to stabilise the system against 'unstructured uncertainty', that is, disturbances on the plant about which there is not enough information to fully model them in the system [8]. These uncertainties, which typically are the result of discrepancies between the modelled dynamics and the true dynamics, can make it very difficult to design a good controller, as there is no simple way to quantify how well the simulated closed-loop behaviour will be reflected in reality. This can result in control systems working very well in simulation, and then failing in implementation. It follows that the ability to optimise for this flexibility against uncertainty - robustness - gives H_∞ control an inherent advantage over many other methods. As mentioned in III-E, robustness is particularly important in this system as the unconventional design configuration limits the ability to accurately model aerodynamic effects.

The robust stabilisation problem can be solved with the Glover-Macfarlane method, which models the uncertainty as perturbations on a normalised coprime factorisation of the plant transfer matrix, and then applies the optimising algorithm detailed in [8]. Expressing perturbations in this form, as opposed to applying a multiplicative or additive change to the matrix, can be shown to be much more effective when the nature of the perturbations is unknown.

This method can be extended to H_∞ Loop Shaping Design Procedure (LSDP), by applying weighting matrices before (pre-compensator) and after (post-compensator) the controller. These matrices refine the effects of the controller on the plant in order to improve performance - since this is equivalent to bending the closed-loop Nyquist curve into a more favourable form, the technique is known as loop-shaping. There are three key performance characteristics which can be changed by loop shaping:

- Reference tracking - the tracking error can be reduced so that the airship better matches its desired trajectory;
- Load disturbances - these can be attenuated so that the unstructured uncertainty on the system is reduced;
- Sensor noise - this can be suppressed to minimise error in the state estimations.

The effect of the weights on these properties can be assessed by plotting the singular values of the plant matrix over a range of frequencies. For a given frequency, the maximum singular value represents the maximum gain of the plant, and so this plot illustrates whether the plant has an amplifying or attenuating effect at a particular frequency. To achieve the desired effects, the contributions of the three properties to the overall tracking error must be described. A well-known result is shown in (30), where P and K are the plant and controller in the frequency domain (as transfer matrices), and $\bar{e}(s)$, $\bar{r}(s)$, $\bar{d}(s)$ and $\bar{n}(s)$ are the respective Laplace transforms of the tracking error, reference signals, load disturbance and sensor noise [9].

$$\bar{e}(s) = \frac{1}{1+PK}\bar{r}(s) - \frac{P}{1+PK}\bar{d}(s) + \frac{PK}{1+PK}\bar{n}(s) \qquad (30)$$

The transfer matrix P describes the open-loop plant, and the closed-loop plant is given by $L = PK$. By inspecting the coefficients in (30), the contributions of $\bar{r}(s)$ and $\bar{d}(s)$ to the error will be suppressed if the gain of the closed-loop transfer matrix is large, and the effect of $\bar{n}(s)$ is reduced when the gain is small. Since references and load disturbances tend to be low frequency signals, and sensor noise only appears at high frequencies, the system should be shaped so that the singular values of L are high at low frequencies, and low at high frequencies.

By combining the robust stabilisation method with loop-shaping, a controller can be designed which has good performance characteristics without sacrificing robustness.

E. Solutions

The design procedure was implemented in MATLAB to obtain controllers for both the FBL model and the RFBL model. The

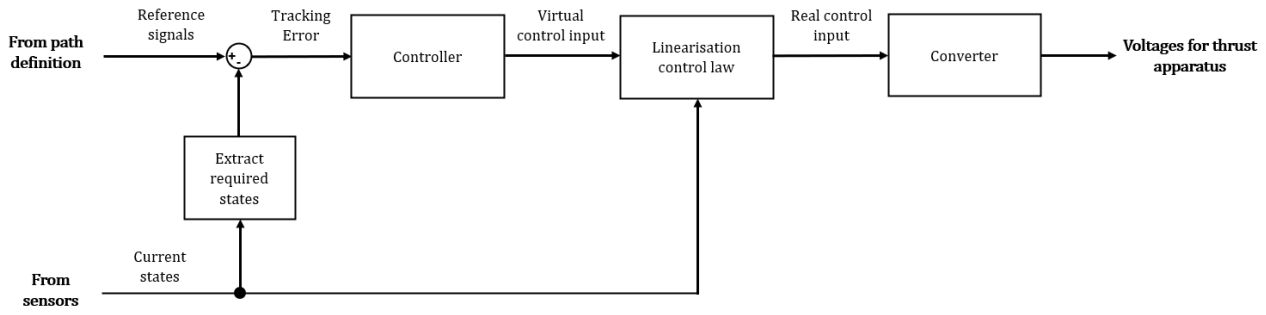


Fig. 3: System architecture

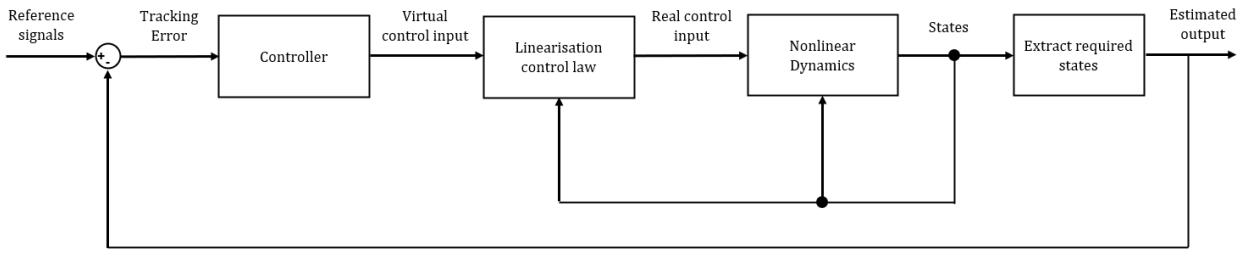


Fig. 4: The simulated system

optimisation was attempted with two different functions: *loopsyn*, which shapes the system to a user-specified target plant and then solves the robust stabilisation problem; and *ncfsyn*, which requires the user to set the weights and then solves for the controller.

For simple systems, such as single input, single output configurations (SISO), loopsyn is preferable as it is relatively straightforward to set a target plant - this can be a scalar transfer function with performance characteristics matching those required by the system. However, it is much more challenging to choose a target plant for a large MIMO system like the airship. Figure 5 shows the singular values for the FBL system, the controller and the resulting shaped plant when *loopsyn* is applied. The target plant was set as a simple second-order scalar transfer function which was known to have good performance characteristics [10]. However the plot shows that this target actually shaped the original model away from the desired form. This is particularly noticeable in low frequencies, where the gain of the shaped plant is below the original - this implies that, to maximise the stability margin, the tracking performance has worsened. Several other target plants were tested, but none were able to improve performance.

For *ncfsyn*, the compensators were chosen by applying the weight optimisation algorithm proposed in [11]. The MATLAB function could then to implemented to find a controller for the FBL system. The singular value plots for this design are shown in Figure 6. The shaped plant has a higher gain at low frequencies than the original system, and the gain has decreased at higher frequencies. Therefore this controller can improve all three performance characteristics. Furthermore, the closed-loop system has a relatively good ϵ_{max} value of 0.3715. For these reasons, this controller was chosen to be applied to the FBL system.

Both techniques were also attempted on the RFBL system, however the state space matrices for this model are considerably

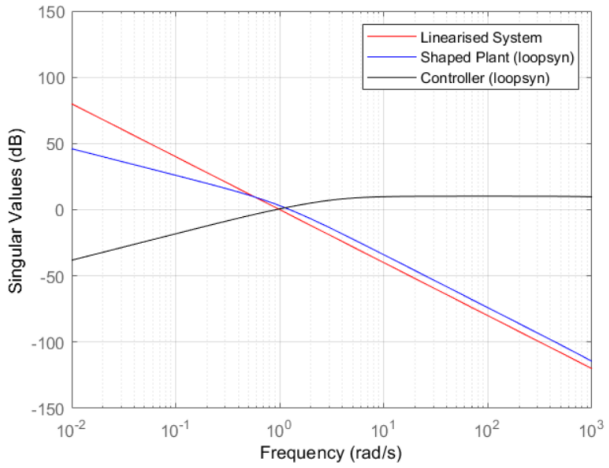


Fig. 5: Singular values using MATLAB *loopsyn* function on the FBL system

more complex than in the FBL system, and neither algorithm was able to run to completion, so it was not possible to obtain a controller for the more robust model. There is no reason why it should be harder to control this model - the failure of the method was purely due to the highly intricate computational procedures required. Figure 7 illustrates the problem: the singular values of the RFBL plant are much smaller than in the FBL model, so very significant loop shaping would be required to match with the approximate target design - clearly it was not possible for the program to achieve this whilst simultaneously optimising robust stability.

Naturally it would have been preferable to find a controller with the RFBL model, but the fact that H_∞ control optimises for a robust closed-loop system suggests that the design will still be

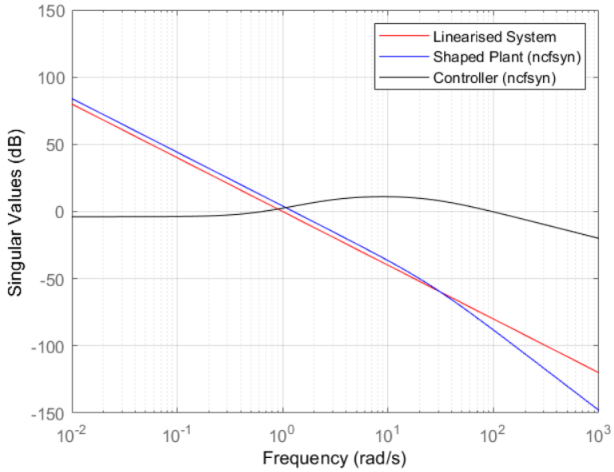


Fig. 6: Singular values using MATLAB *ncfsyn* function on the FBL system

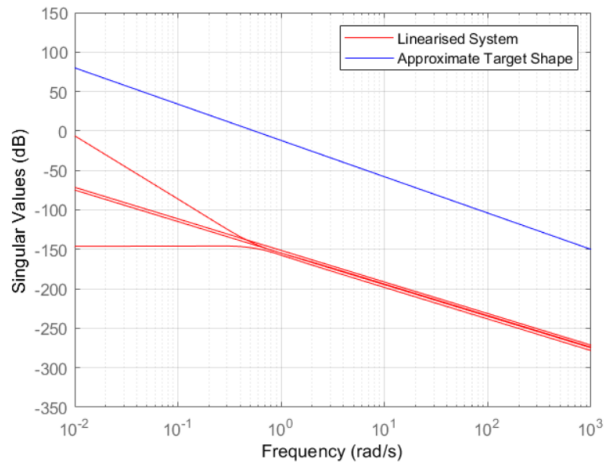


Fig. 7: Singular values of the RFBL system

able to perform well when uncertainty is applied.

V. Simulation Results

A. Constructing the Simulation

The performance of the controller was assessed by running a series of simulations on Simulink. The Simulink model followed the structure of the system in Figure 4, but was extended to also give the required actuator signals, as illustrated in Figure 3. Saturation limits were imposed on the propeller rotor speeds and motor tilt angles to reflect the operational range of the devices. The propellers cannot exceed $88000\mathrm{RPM}$ and cannot change direction, so the speed must always be a positive number. The tilt angles must range between $-90°$ and $+90°$.

The system was tested in different manoeuvres by defining trajectories for the airship to follow. These trajectories could be set by specifying checkpoints along the path, and then applying a MATLAB function to generate a time series of reference positions and heading (yaw) angles [12]. When deriving the feedback linearisation model, the vector of measured outputs was defined as $\boldsymbol{y} = \begin{bmatrix}\boldsymbol{\eta}^T & \boldsymbol{\lambda}^T\end{bmatrix}^T$. This implies that the reference signals should be the three position coordinates and the three orientation angles of the airship, and so reference values for roll and pitch must also be selected. These two signals should always be fixed at zero, as this sets the airship level, thus making the sensor readings more reliable.

B. A 2-D Trajectory Test

Initially, only trajectories in the $x-y$ plane were tested. The first priority was to establish whether the airship can manage turns along a path. The asymmetric layout of the propulsion systems, and the reliance on three tilting motors to offset the need for rolling, provide a number of challenges when attempting such manoeuvres. The first such trajectory is shown in Figure 8a, featuring a rounded turn and a sharper turn. The airship is able to track the initial straight section well, but very quickly fails as it enters the first turn. Figure 8b shows that, as soon as the airship starts travelling, it pitches down dramatically, whilst simultaneously rolling to the left and yawing to the right (recall the sign conventions shown in Figure 1). By the time it enters the turn, the orientation is already very far from the reference, so the change in the path forces the controller to attempt to correct very large tracking errors. The required actuator inputs to achieve this lie well beyond the saturation limits, as shown in Figure 8c, so the airship spirals out of control.

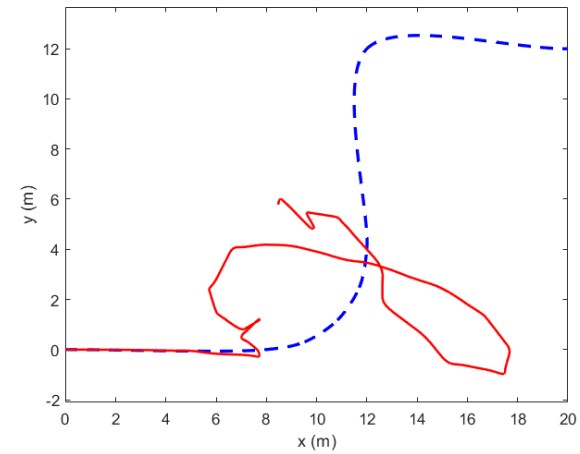
By adjusting the parameters of the system and running more simulations of the same trajectory, it was clear that this poor performance was due to the airship being almost neutrally buoyant - the weight was less than 1% larger than the upward buoyancy force opposing it. This means that even a very small component of thrust acting in the positive z_E direction is enough to cause a resultant upward force on the airship. To avoid this, the motors will immediately tilt to their $90°$ limit at the start of the trajectory, as seen in Figure 8b. Such abrupt control inputs will cause tracking errors, as there is a lag time between reference signal iterations and physical changes to the actuators. This causes the tilt angles to fluctuate, and so a resultant upward force is generated and the airship starts to rise. The combination of an almost-negligible weight excess combined with an inability to set downward thrust components makes it virtually impossible for the airship to correct this altitude error unless it pitches down. By pitching, the thrust vectors can be redirected without exceeding the tilt limits, but the controller will then attempt to correct the non-zero pitch and the system will diverge to an uncontrollable state.

Having established the severe limitations of such a small weight excess, the target mass of the airship was increased so that a 6% weight excess could be achieved. This value is used in all subsequent simulations.

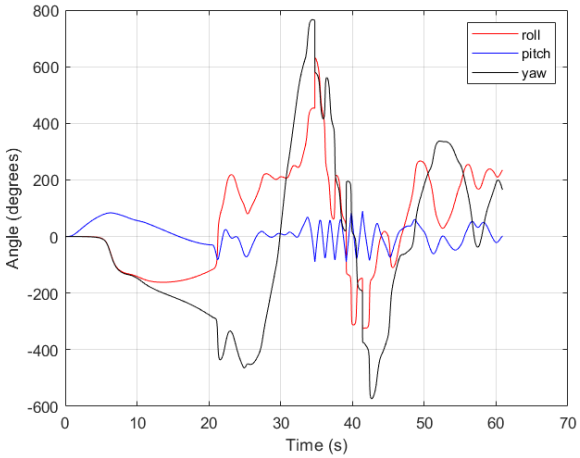
The initial trajectory test was repeated with the new weight excess. Figure 9a shows that the airship is now able to follow the path very well to completion. Other than a very small pitch-down at the start of the trajectory, which is due to the sudden acceleration from a stationary starting point, the vehicle maintains perfectly level flight throughout, as shown in Figure 9b. Figure 9c illustrates the difference in motor inputs for the rounded turn (starting at around $15\mathrm{s}$) and the sharp turn (around $43\mathrm{s}$). The rounded turn is performed with a series of small pulses on the actuators over a $15\mathrm{s}$ period, whereas the sharp turn requires a single large pulse. Although the tilt angles are considerably larger, the thrust requirements for the sharp turn are very modest. This suggests that it is not necessarily less energy-efficient than a rounded manoeuvre - this is a useful result as it implies that the airship can cope with abrupt changes to its trajectory.

C. Vertical Takeoff and Landing

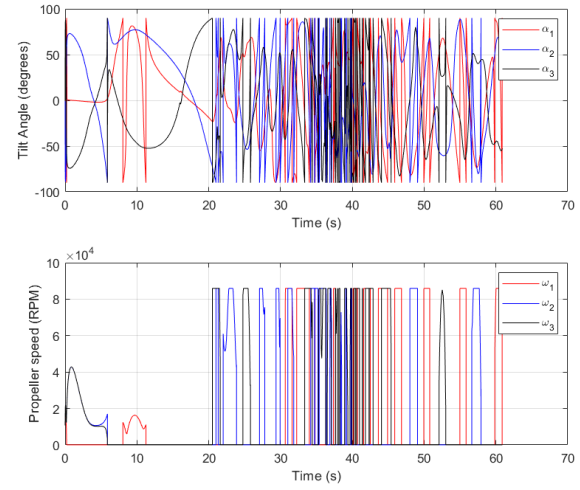
The airship should be able to takeoff and land vertically. The ability of the control system to manage these manoeuvres was simulated with a vertical ascent to $2\mathrm{m}$ altitude, followed by



(a) Trajectory tracking

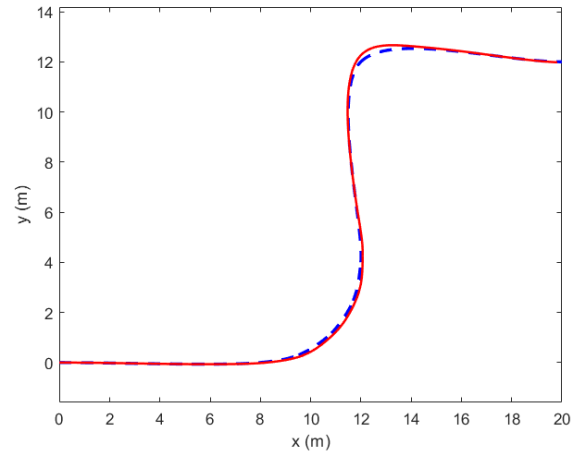


(b) Airship orientation

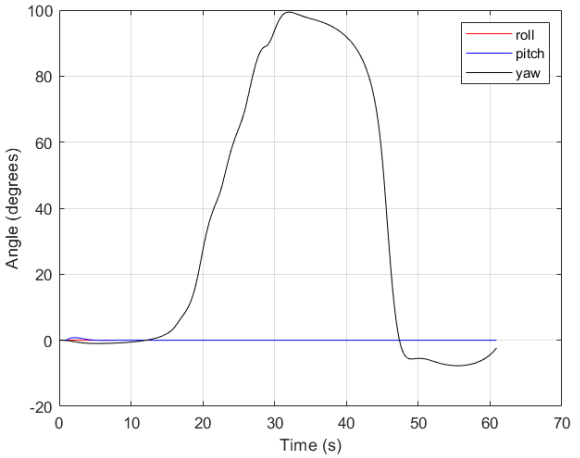


(c) Propeller speeds and motor tilt angles

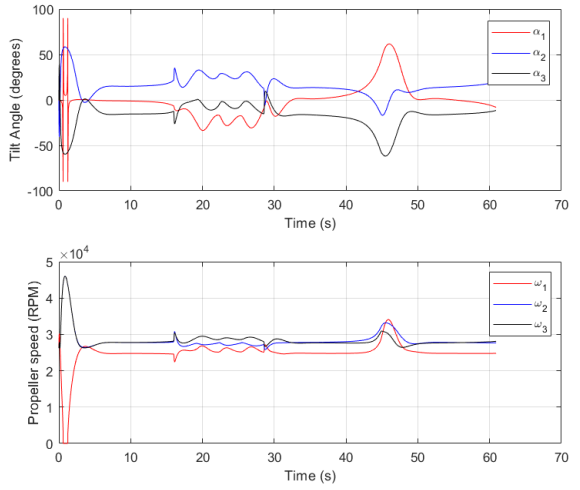
Fig. 8: Simulation results for an initial horizontal trajectory.



(a) Trajectory tracking



(b) Airship orientation

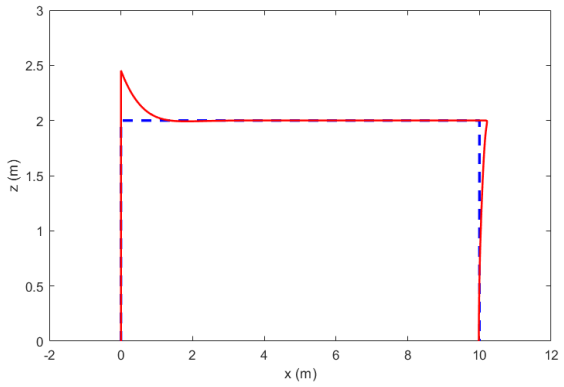


(c) Propeller speeds and motor tilt angles

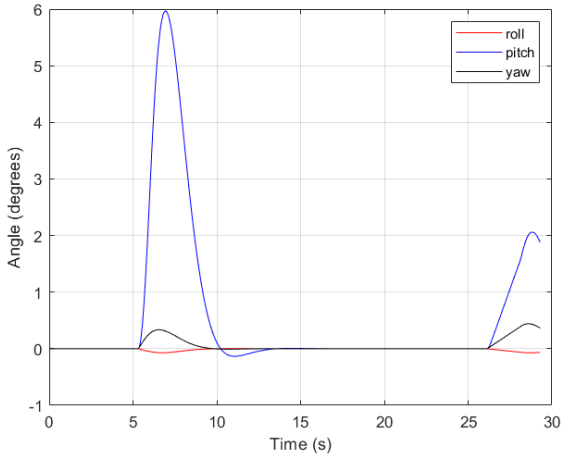
Fig. 9: Simulation results for the initial horizontal trajectory, with the weight excess now increased to 6%.

straight travel over a 10m length, and then finally a vertical descent back to the ground. The desired path and the airship trajectory are shown in Figure 10a. There is a vertical overshoot of almost 50cm at the start.

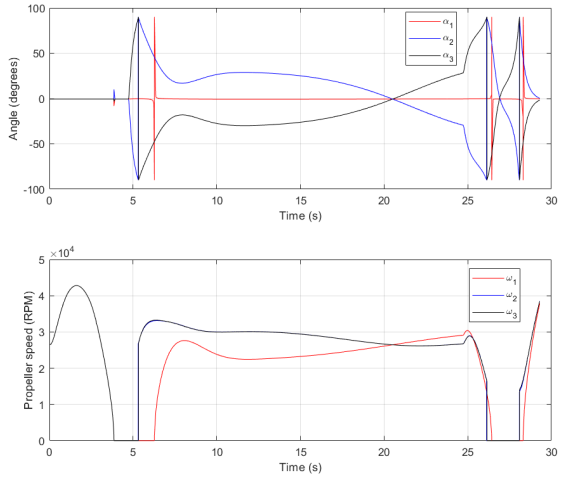
From the actuator signals shown in Figure 10c, it is clear that this overshoot is the result of a large initial thrust input, required to lift the airship off the ground. Once the airship has passed the new reference altitude, it can only drop to 2m by setting all thrust values to zero, and sinking purely with weight excess. While this may suggest that an even higher weight excess would cause the altitude to settle in a shorter time, this has to be compromised with the fact that a higher weight excess would also cause a higher initial overshoot, as larger actuator impulses would be required at the start of the trajectory. Furthermore, minimising weight excess to improve energy efficiency is more critical for this design than



(a) Trajectory tracking



(b) Airship orientation



(c) Propeller speeds and motor tilt angles

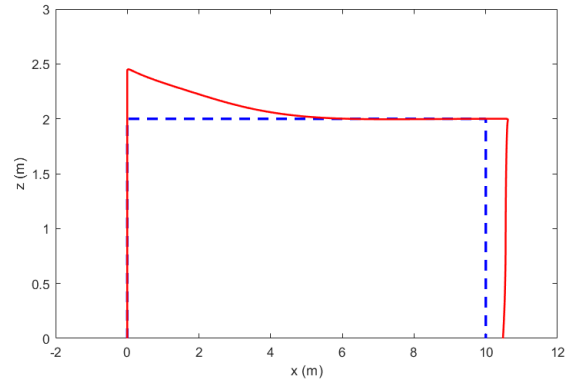
Fig. 10: Simulation results for a vertical takeoff, straight travel and vertical landing.

reducing overshoot in takeoff - the priority should be to keep actuator inputs as low as possible to reduce power consumption and extend the mission endurance, rather than always striving for the shortest path.

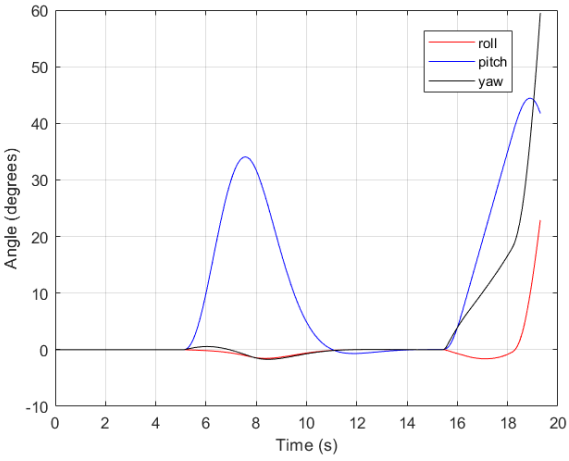
To help the airship settle at the correct value, it briefly pitches down very slightly, but then it completes the remainder of the horizontal path while staying perfectly level and straight, as seen in Figure 10b. As it approaches the descent section, Figure 10c shows that the airship decelerates by tilting the front two motors so that they generate a thrust component which opposes the direction of travel. The slight horizontal overshoot forces the tilt angles to increase to the saturation limit - in this way, the horizontal error is corrected without causing a vertical error. As the airship starts to descend under its own weight excess, the propellers start to speed up again in order to decelerate the fall.

D. Operational Speed

In all the simulations so far, the airship has not travelled faster than $1\mathrm{m/s}$. However the design was originally intended to be able to reach $1.5\mathrm{m/s}$. In the takeoff and landing trajectory shown in Figure 10a, the airship accelerated to a maximum speed of $0.7\mathrm{m/s}$. By adjusting the inputs into the trajectory function, the same path can be simulated for a maximum speed of $1.5\mathrm{m/s}$, with the result shown in Figure 11a. Since the system now requires greater acceleration, the thrust inputs will be larger. However it has already been established that increased thrust can cause a resultant upward force on the airship. It follows that the altitude will settle to the reference value over a larger distance, but also the vehicle will have to pitch down much more to achieve the necessary speed whilst descending. Figure 11b shows that the airship now reaches a $35°$ pitch in order to follow the set path. This is potentially problematic, as such an extreme orientation will render several of the sensors unreliable, thus making it very difficult to enforce any abrupt changes to the trajectory, such as an emergency stop.



(a) Trajectory tracking



(b) Airship orientation

Fig. 11: Simulation results for vertical takeoff, straight travel and vertical landing, with maximum speed now increased from $0.7\mathrm{m/s}$ to $1.5\mathrm{m/s}$

Even once the airship has settled at the desired altitude, its speed is so high that the horizontal overshoot at the start of the

descent cannot be corrected. The high actuator inputs required to reduce this tracking error cannot be achieved due to the saturation limits, so the airship rolls, pitches and yaws significantly as it descends.

Clearly the speed of the airship has to be strictly limited for certain manoeuvres. In theory, it can reach $1.5\mathrm{m/s}$, but this can only be achieved safely if the acceleration occurs over a large distance. For the simulation in Figure 11a, the speed increase was completed in $5\mathrm{m}$, leaving $5\mathrm{m}$ more in which to decelerate to rest. Further simulations showed that, if the available distance is approximately doubled, the airship can reach the target speed without excessive pitching. In reality, it is highly unlikely that the navigation algorithm could map a straight, uninterrupted path of $20\mathrm{m}$, suggesting that the target operational speed of the design should be reduced.

E. 3-D Trajectory Simulation

From the previous tests, it is apparent that the ability of the control system to negotiate turns, takeoffs and landings is dependent on the flight speed and the design of the trajectory. In order to better characterise the nature of these dependencies, and how they collectively influence performance, a number of simulations were run for 3-D trajectories with turns and vertical motion, applying different speed constraints to assess the system's limitations. The tracking tended to fail due to the airship's orientation becoming unstable in response to large control inputs. It was observed that, when entering a turn with too much speed, the combination of the airship pitching down and the controller attempting to match a yaw reference would result in a large roll.

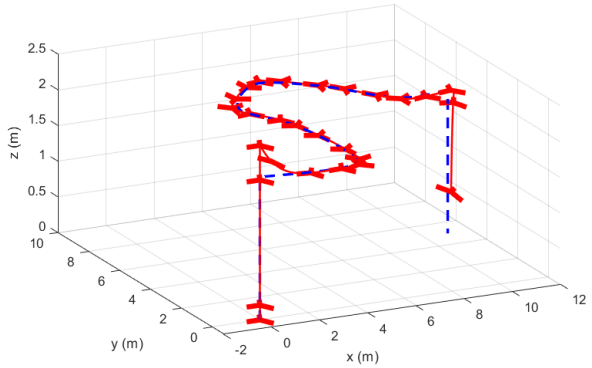
The coupling of these three angles was visualised by extending the trajectory plots into animated simulations on MATLAB. The airship was represented as a tricopter, since this made the orientations very clear. The model was animated by defining the motor positions relative to the centre of the body, and then using Euler angle transformations on the current states to plot the moving shape.

An example test is presented here, showing the visualisation of the airship dynamics for the same trajectory under a number of different speed constraints. The simulations ran as smooth animations, but to illustrate that effect they are presented here as time lapse stills along the trajectory.

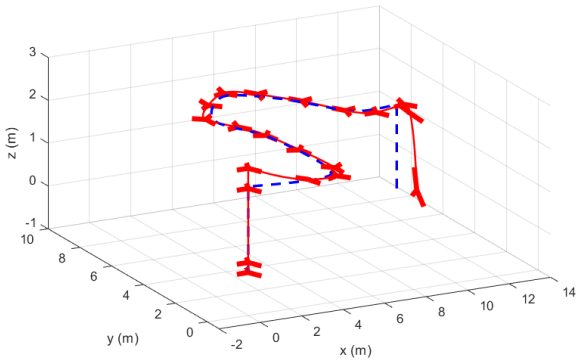
The defined path begins with a vertical ascent, then follows a curved route before ending with a vertical descent. The tracking performance was tested for three different maximum speeds: $0.6\mathrm{m/s}$, $0.9\mathrm{m/s}$ and $1.2\mathrm{m/s}$. The resulting time lapse plots are shown in Figure 12. For the slowest case, the controller is able to achieve almost no tracking error other than an initial overshoot. However in the vertical descent the airship starts to pitch down as it falls. This is because the turn just beforehand caused the body to overshoot the required yaw angle, so the descent begins when the airship incorrectly oriented. Since descents initially begin with free fall due to weight excess, the motors attempt to correct the orientation without generating an upward decelerating component. This causes the slight pitching as the airship approaches the ground

For the $0.9\mathrm{m/s}$ case, there is greater tracking error throughout the path. This is triggered by the airship overshooting the first turn following an extended pitch-down acceleration. There is some overshoot on the second turn too, but the most noticeable difference is that, since the landing phase now begins with an even greater yaw overshoot, the extent of the pitching in descent is much more dramatic. By the time it reaches the ground, the pitch angle has increased to $80°$.

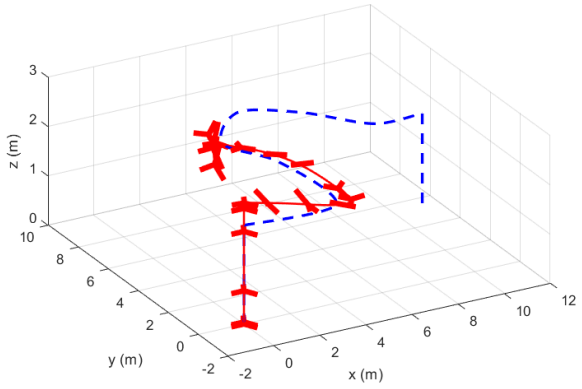
When the maximum speed is set to $1.2\mathrm{m/s}$, the airship fails to complete the path. The speed before the first turn in fact rises to $1.3\mathrm{m/s}$, so the resulting actuator signals have to be very large, thus causing the vehicle to yaw too much. The tracking error of the heading angle cannot be corrected before the next turn, so the tilt angle signals become saturated, causing the airship to roll and pitch out of control.



(a) Maximum speed: $0.6\mathrm{m/s}$



(b) Maximum speed: $0.9\mathrm{m/s}$



(c) Maximum speed: $1.2\mathrm{m/s}$

Fig. 12: Time lapse plots of the airship tracking a sample 3-D trajectory

Similar tests can be carried out for other 3-D trajectories, which can then be used to refine the control system without having to change the airship design itself. Some possible methods for doing this are proposed in the following section.

F. Improving Control Performance

As described in V-B, the control performance was significantly improved by increasing weight excess. This is a clear general trend, so adding further mass to the design would enable more

reliable turns and larger operational speed. However this would result in the mission endurance being reduced as the control system would become less energy-efficient - this is because larger thrust inputs would be required to maintain level flight. Furthermore, a heavier design may be more sluggish in turns, so larger tilt angles will be required to manoeuvre the airship. Even if the resulting tracking is very good, this is yet another factor that will increase power consumption and reduce endurance.

Instead of attempting to improve performance by changing the physical design, the trajectories themselves can be defined in such a way that the physical characteristics of the airship and the saturation limits of the actuators are taken into account. This idea is introduced in [12], where vertical overshoot is suppressed by shaping the trajectory signals to force a deceleration as the target altitude is approached. The same concept can be applied to horizontal manoeuvres.

As the simulations demonstrated, if the reference signals demand too large an acceleration, not only will the airship pitch down excessively, but also the tracking is likely to fail at the next turn or stop. Therefore, instead of just defining the trajectory in terms of positions, maximum speed and heading angle, the reference signals can also be adapted to depend on a a threshold acceleration. In this way, if a large maximum speed is requested over a short distance, the acceleration limit can overwrite this command with a safer value.

Additionally, the limitations of the closed-loop system vary greatly depending on the complexity of the desired path. Further simulations showed that, for higher speeds, the airship can still manage rounded turns but will fail at sharper turns about a point. This is because the turn is performed with a step input on the yaw signal, so the actuators initiate a sudden swivel which will generally overshoot the desired turn angle. This effect worsens as the turn angle increases because the magnitude of the input is larger. This issue can be solved by defining a yaw trajectory for sharp turns - this can be expressed as the integral of a maximum yaw rate.

These trajectory shaping methods can always be extended further by running more simulations of different possible manoeuvres. For each type of problem that arises, a new condition can be added to the trajectory function. Then any path required by the navigation algorithm can be translated into a set of reference signals which define the path in the most manageable way possible.

VI. Comparison of the Controllers

The simulation model was adapted to test the Model Predictive Control (MPC) and Linear Quadratic Regulator (LQR) controllers in comparison to the H_∞ design. These tests are detailed respectively in [13] and [14]. As described in [12], Neural Predictive Control (NPC) was ruled out due to the high complexity of implementation.

Considering the tracking error alone, MPC has the best performance - this is to be expected, as it is the most advanced of the three methods. Even so, LQR and H_∞ controllers both give relatively good tracking as well. Despite the differences in control design, all three faced the same limitations in speed and acceleration due to the low weight excess. However, as discussed in the previous section, the most efficient way to mitigate this problem is by refining the reference signal definitions, instead of amending the controller or the physical design. Considering also that the airship's mission profile does not demand high speed or acrobatic manoeuvres, optimal tracking performance is not necessarily the most critical design factor.

Since all controllers provide reasonably good performance, robustness is arguably the key objective for the closed-loop system. This is due to the complex arrangement of the actuators, which reduces the allowable margin for error in the control system, combined with the physical uncertainties of the dynamics model. Since this particular design configuration has not been studied in detail before, there is no available empirical data to aid the model, so ideally a prototype should be tested in a wind tunnel in order to improve the mathematical description of the dynamics, and hence the reliability of the control system. Assuming this option is not possible, the use of a sufficiently robust controller is imperative. Out of the three candidate controllers, the H_∞ design detailed in this report is the only one which optimises for robustness, whereas the other two prioritise performance. However, it should also be considered that LQR and MPC were designed with the more robust dynamics model, whereas an H_∞ controller could only be computed for the FBL system.

For all the points in favour of each controller, ultimately the choice has to be restricted by its suitability for integration on the Raspberry Pi. The Pi on board the airship has $1\mathrm{GHz}$ of processing power, and it is required to run the control system as well as all the navigation code, including location mapping and path finding instructions. This may well rule out MPC, as it calculates a new controller matrix at each iteration of the closed loop. This makes it much more computationally expensive than the other two designs, where the controller is fixed.

At the time of completion of the project, several of the navigation codes required to run on the Pi were yet to be developed, so selecting a definitive controller out of the three candidates was not feasible.

VII. Conclusion

A full design iteration of the airship control system was carried out, starting with an analysis of the flight dynamics, then applying Feedback Linearisation to create a model which could then be used to design a controller with the H_∞ Loop Shaping Design Procedure. A Simulink model was constructed to simulate the performance of the controller for various reference trajectories, and this was used to create an animated simulation on MATLAB to visualise the behaviour of the system when following more complex paths.

While the control of this system is made significantly more difficult by the large buoyancy force opposing the weight, as well as the unconventional propulsion system design, it was concluded that the performance of the airship can be significantly improved with only a slight mass increase, and any other limitations can be addressed by defining more versatile reference signals which better reflect the nature of the system.

Comparison with LQR and MPC designs illustrated the need for a compromise between robustness, performance and computational complexity. Since the proposed trajectory shaping methods provide a number of ways for the performance to be enhanced, the other two properties are considered more critical when choosing which controller to implement. There was insufficient information to accurately quantify the restrictions on computer processing power, but, assuming that implementation is not an issue, the simulations suggested that all three controllers would give satisfactory results.

REFERENCES

[1] L.M. Nicolai G.E. Carichner. *Fundamentals of Aircraft and Airship Design, Vol.2*. American Institute of Aeronautics and Astronautics, 1992.

[2] E. Papadopoulos M. Ramp. On modeling and control of a holonomic vectoring tricopter. *IEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[3] A. Lanzon M.K. Mohamed. Design and control of novel trirotor UAV. *Proceeding of the 2012 UKACC International Conference in Control*, 2012.

[4] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 1995.

[5] E.R. De Pieri H. Guillard A.L.D Franco, H. Boules. Robust nonlinear control associating robust feedback linearisation and H_∞ control. *IEEE Transactions on an Automatic Control, vol. 51, no. 7, pp. 1200-1207*, 2006.

[6] A. Lanzon J. Hu. Cooperative control of innovative tri-rotor drones using robust feedback linearisation. *Proceedings of the 12th UKACC Internation Conference on Control, Sheffield, UK*, 2018.

[7] B.A. Francis. *A Course in H_∞ Control Theory*. Springer-Verlag, 1987.

[8] D. C. Mcfarlane K. Glover, J. Sefton. A tutorial on loop shaping using H_∞ robust stabilisation. *IFAC 11th Triennial World Congress, Tallinn, Estonia*, 1990.

[9] R.M. Murray K.J. Astrom. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2009.

[10] Mathworks. Robust control toolbox for use with MATLAB. 2001.

[11] A. Lanzon. Weight optimisation on H_∞ loop-shaping. *Automatica, Vol. 41, No. 7, pp. 1201-1208*, 2005.

[12] Jeffrey Yam. Autonomous airship design individual report - flight dynamics and control. Technical report, Department of Aeronautics, Imperial College London, 2020.

[13] Bianca-Lorena Giocas. Autonomous airship design individual report - flight dynamics and control. Technical report, Department of Aeronautics, Imperial College London, 2020.

[14] Nikhil Purmessur. Autonomous airship design individual report - flight dynamics and control. Technical report, Department of Aeronautics, Imperial College London, 2020.