Nonlinear Control and State Estimation of Holonomic Indoor Airship

Yin Yang

Department of Mechanical Engineering
McGill University, Montreal

August 2011

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of

Master of Engineering

Abstract

Three full-state optimal controllers are proposed to fulfill the requirements of flying an indoor holonomic airship in real-time, namely, hovering control, set-point control and continuous reference tracking. In the hovering control design, the airship is assumed to be a quasi-stationary plant, and an infinite horizon linear quadratic regulator (LQR) operating in a gain scheduling manner is employed. Meanwhile, a controller based on the state-dependent Riccati equation (SDRE) and ad hoc feedforward compensation is synthesized to tackle the set-point control problem. Lastly, a continuous tracker is dedicated to rejecting all disturbances along any given reference trajectory. With reasonable computation cost, the proposed controllers show significant advantages over the PD controller in both simulation and real flights.

A state estimator designed with the unscented Kalman filter is also implemented in this work. The purpose is to track the airship state for the feedback loop and other navigation tasks by fusing information from the on-board (an inertia measurement unit and a laser range finder) and/or off-board (an infra-red based motion capture system) sensors. A loosely coupled sensor fusion scheme is employed and validated in experiments.

Abrégé

Trois méthodes optimales de commande à retour d'état complet sont proposées ici afin d'accomplir les exigences du vol intérieur d'un ballon dirigeable holonomique, et ceci, en temps réel. Les manoeuvres exigées incluent le maintien d'une position stationnaire, le mouvement vers un point et suivant une trajectoire continue. Pour la régularisation, un modèle quasi-stationnaire du ballon est assumé et un régulateur quadratique-linéaire (LQR) à horizon infini est utilisé dans un mode d'échelonnage des gains. De plus, les mouvements vers un point sont accomplis en se basant sur le retour d'état pour résoudre l'équation de Riccati qui en dépend et pour compenser la dynamique non-linéaire. Finalement, les perturbations autour d'une trajectoire continue sont rejetées par une méthode dédiée afin de suivre cette trajectoire. Preuves expérimentales et simulées à l'appui, ces méthodes de commande démontrent des avantages significatifs par rapport aux méthodes classiques de commande porportionelle-dèrivée (PD), et ceci, avec des exigences modérées sur le système informatique.

Ce travail de thése démontre aussi l'utilisation d'un filtre de Kalman non-parfumé (UKF) pour estimer l'état du système. Cette estimation produit le retour d'état complet nécessaire aux méthodes de commande et à d'autres tâches de navigation en combinant les mesures de différents systèmes enbarqués (système inertiel et télédétecteur par laser) et non-embarqués (système de capture du mouvement à l'infra-rouge). Une méthode de fusion sensorielle à séparation partielle est utilisée et validée expérimentalement.

Acknowledgments

First, I sincerely thank my supervisor Prof. Inna Sharf for her tremendous help and guidance throughout my Master's program, especially the time and effort she devoted to improve my writing. Her financial support is also critical for me to study at McGill University. Next, I would like to thank Prof. James Forbes for his advice on the model discretization and attitude estimation part of this work. My appreciation also goes to everyone at the Aerospace Mechatronics Laboratory and in particular Mikelis Valdmanis for his exploration of our indoor airship as an unmanned aerial vehicle. Part of this work is motivated by his early-stage exploration and the flights we carried out together. Also, I would like to thank Mikael Persson for providing a French translation of the abstract of this thesis. Lastly, I want to thank my parents and family for everything that they have done for me, and Jiandi Yao for her unwavering support. Without them, this work can not be accomplished.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background on Indoor Airship

The indoor airship discussed in this thesis was fully developed at the Aerospace Mechatronics Laboratory (AML) at McGill University. The purpose of having this airship was to emulate a free-floating satellite for research in space robotics. The novel design of the airship not only gave an opportunity to experimentally study gravity-free systems on Earth, but also ignited our interests in the airship as a stand-alone platform. The airship, as a stand-alone unmanned aerial vehicle (UAV), mainly benefits from its lighter-than-air characteristic. Theoretically, it needs no energy to maintain neutral buoyancy when there is no air flow, making it an interesting platform for long term indoor surveillance. In addition, the airship has the capability for holonomic motion in six degrees-of-freedom without any one direction being the main motion axis. In contrast, the main motion axes of a quad-rotor vehicle are the three translational axes and the yaw axis. Pitch and roll motion can only be achieved to small angles. This gives the airship more motion flexibility than is the case for most UAVs at present.

Initially our focus was on improving the functionality of the airship by adding a rigid frame and developing a stable Proportional-Derivative (PD) controller for it [56]. A commer-

cial motion-tracking system was installed in the AML to track the position and orientation of the airship. (Details about the airship design and the sensing techniques will be provided in the following sections.) To follow the idea of developing the airship as an autonomous UAV with potential applications such as indoor surveillance, some of the previous work focused on localizing and navigating the airship [63] based on on-board sensors, including a low cost camera, an inertial measurement unit(IMU), and a laser range finder (LIDAR). Our ultimate goal is to be able to fly the airship autonomously through any space, ideally, without any preparation or previous knowledge of the environment. To demonstrate the full capability of the airship while still tackling a realistic scenario, we define the following goal: to fly the airship up a stairway without colliding with any obstacles. This may not be a real challenge for some micro-UAVs, but for an airship of 6 ft in diameter, limited thrusts and 3.6 kg distributed mass, this brings the problem of pose and velocity control to a new level of difficulty. Meanwhile, a good feedback system needs accurate state estimation. Generally, robots are unable to navigate in space accurately without querying their current state. Therefore, as a prerequisite to achieving our goal, we need to solve the control and estimation problems.

1.2 Problem Specification

With the ultimate goal and the status quo of the airship laid out, the primary challenge is clear: for the maneuvers needed in the indoor navigation, design controllers which outperform the existing PD controller, but also maintain robustness in terms of handling disturbances and unmodeled error. Since the airship can be viewed as a rigid-body with 6 DOF (degrees of freedom), the designed controllers should also resolve a general set of problems related to pose and velocity regulation of a rigid body. Moreover, the proposed controllers have to be practical under a hard real-time constraint, since in a real flight, delay of a fraction of a second in command signal can result in poor response of the airship. For the controller to achieve

its performance objectives, a state estimator is needed for the full-state feedback system. It will not only provide reliable estimation of the state, but also combine the information coming from various sensors, either on-board or off-board, with different measurement models and update rates. With the successful implementation of the controllers and the estimator, we hope to eventually not have to use the off-board sensors, and to achieve a stand-alone autonomous aerial platform.

1.3 Review of Related Work

According to the problem specification, a survey of system design and control/estimation strategies of the existing UAVs has been performed. This section is going to present the literature review in three subsections: the first focuses on indoor airship design and its applications; the second part gives successful examples of UAV control following the order of different control algorithms; the last part summarizes some research in UAV state estimation field.

1.3.1 Indoor Airship

Over the last few decades, indoor airships (blimps) have attracted interest in various contexts. Most of the existing airships are simply miniature versions of the traditional airships: ellipsoidal in shape with thrusters at the rear and fins for stabilization. 2q In [70], an indoor airship named Blimp2b is introduced to the visual servoing community. The airship consists of three 8 mm (in diameter) DC thrusters, a forward-looking camera, a rotation anemometer and a MEMS based gyroscope measuring yaw rotation. The authors of [70] developed a comprehensive dynamics model of their airship, and it is presented together with a pragmatic methodology for parameter identication. A classical vision-based navigation scenario (2D course stabilization and obstacle avoidance in randomly textured environment) is used to demonstrate the effectiveness of their approach. Recently, the same authors

consider evolutionary control of the airship [69], using a so-called spiking neural circuit to connect the vision system and the motors. Genetic algorithms are employed to evolve the architecture of the circuit, by learning the interaction dynamics between the airship and the test environment.

As a novel flying display system, a small autonomous blimp along with a visual tracking system and an image projection system is presented in [48]. The visual tracking system tracks the blimp while it flies along a given spatial path to follow a wall. The projection system projects images which have been corrected to look natural on the ellipsoidal surface of the blimp, and the blimp is designed with holonomic dynamics. It can maintain a stable pose in the presence of bounded air flow disturbances during the wall following motion. In that research, a simple dynamics model is decoupled and utilized to design a PD controller.

Stepping away from the traditional airship shape, Kang et al. [36] design an indoor airship as a robotic agent for tele-presence. Their innovation is the hybrid structure which combines a blimp and a wheeled vehicle. The airship is able to move over ground obstacles conveniently, and to maintain the standing phase on the floor during communication as a ground station. The main difficulty is to keep a stationary position in standing phase because of the air flow disturbance acting on the blimp. With the 3-DOF dynamics model, PD-based computed torque method is proven to be feasible in both simulation and experiments.

Since indoor airships are usually very limited in their payload capacity, the on-board sensing and power can not be as good as those used on the ground robots or even on other UAVs. Sometimes, design of a flight system for indoor airships can also be difficult due to the uncertainty in the flight (e.g. occasional air flow disturbance) and poor measurement precision. For these reasons, traditional control and estimation techniques are avoided by some researchers. The authors of [13] study the airship maneuvering and control from a biological perspective. The control signal is generated by an intrinsic hierarchical insect based neural model. The visual information captured by two colour CCD cameras on-board is processed independently to extract stabilization or collision cues. The outputs from both

visual streams are projected onto an optomotor group that decides which action to send to the UAV. Their bio-inspired airship is able to complete the basic navigation tasks including course stabilization, drift compensation and collision avoidance.

In [40], Ko et al. study the indoor airship motion by using reinforcement learning. As a non-parametric tool for regression in high dimensional spaces, Gaussian processes (GP) are combined with the dynamics model of the airship to learn the modelling uncertainty during the flight. This combination is proven to be more efficient than just propagating the dynamics equations, and once the parameters of a GP are gained from training, an optimization process is used to find a locally stationary parameter set for the feedback loop.

Another interesting work is found in [68], in which the visual servoing technique is applied to an indoor blimp. With a micro CCD camera on-board, the authors successfully transfer the dynamics parameters of the system, including masses and inertias, into the image plane. By following this track, the explicit computation of pose is avoided. The airship is able to track a slowly moving ball by using PID control first on the image plane, then transferring the control signals to the real thrusters.

1.3.2 Model-based Nonlinear Control and Its Application to UAVs

The approaches for controlling a nonlinear system can basically be classified into two categories: model-based control and model-free control. For model-free control, such as reinforcement learning control and artificial neural network control, we have already seen some successful applications in [40] and [70]. However, the focus of this thesis is on model-based control, since the prior knowledge of the system can directly aid in designing the controller. The airship's performance can be easily adjusted upon the change of any dynamics parameter, whereas the model-free methods may require a complete learning process again. A selection of nonlinear control schemes and their applications to different UAVs, not just the indoor airships, will be reviewed in this section, due to the similarity of the contexts.

Probably the most widely used control theory in the field of UAVs is still PID control, because it is straightforward and easy to implement, even in the absence of knowledge of the plant. A typical application of PID control is found in [37], where the airship maneuver is decomposed into basic motions, such as rotation and straight-line motion. Each motion has an associated PID controller and different motions required for specific tasks are achieved by continuously combining basic motions. The PID control has been proven to work with many nonlinear processes in practice; however, its performance may deteriorate when it comes to UAVs with high nonlinearities. To that end, a number of modified PID controllers have been proposed. For example, Takamasa Sato et al. use Memory-based PID controller for the flight control of an indoor airship [60]. Somewhat like the gain scheduling approach, in this new scheme, PID parameters are automatically generated and tuned based on input/output data pairs of the controlled object stored in the database. Their experiments show that without the proposed modification, the PID controller alone tends to perform poorly if a complex trajectory is prescribed for the airship.

A common approach to alleviate the system nonlinearity is called feedback linearization (FBL), which is to exactly transform a nonlinear dynamic system into a (fully or partly) linear one so that linear control techniques can be applied [28]. There are two techniques in the scope of FBL: one is called input-state linearization where the full state equations are linearized, and the other is input-output linearization where the input-output map from the control $\mathbf{u}$ to the system output $\mathbf{y}$ is linearized. The main limitation of the FBL is that it cannot be used for all nonlinear systems, and both techniques have somewhat stringent conditions to satisfy. In addition, no robustness is guaranteed in the presence of modelling uncertainties. As a successful example of feedback linearization control in the UAV area, in [15], the authors try to use FBL and linear controller to steer a nonlinear helicopter system. Since the model used in the controller design is simplified, not all nonlinearities are cancelled as expected. A neural network (NN) is employed to learn and compensate for the residuals.

Another promising tool for controlling nonlinear systems is adaptive control which

involves a mechanism for adjusting the controller parameters. Usually, an adaptive control system contains two loops: one loop is a normal feedback with the process and the controller, while the other loop is the parameter adjustment loop, executing slower that the feedback loop. Three classical adaptive schemes are gain scheduling, self-tuning regulator (STR) and Model-reference adaptive control (MRAC) [9]. Although there exists relatively little theory for the adaptive control of nonlinear systems, compared with linear systems, some attempts at application to UAVs have been found. For example, in [6], the authors construct an adaptive control law augmented with the variable structure controller, parameter identification algorithm and a tunable pre-filter, for a hypothetical UAV. Their adaptation algorithm is aimed to aid the closed-loop system dynamics in case the parameters of the UAV linear model change in attitude control.

Sliding mode control (SMC) has attracted a lot of interest in the recent years. This control scheme can be divided into two steps: First, a lower dimensional (lower than dimension of the system) manifold is found which can be stabilized and made positively invariant by an appropriate control input. Then, a second control is chosen to force the trajectories, which are not initialized on the manifold, to converge to the manifold within finite time. SMC has been shown to achieve robust performance by effectively accounting for parameter uncertainties and unmodeled dynamics. The drawback is that it results in discontinuous high gain controllers with ensuing chattering of control. In [58], SMC is used in the scenario of multiple UVAs separation trajectory control. The design in [58] turns out to be effective in simulation when the wing aircraft is affected by the unmodeled vortex of the adjacent lead aircraft.

Finally, we move on to the optimal control, which is well established for linear systems and has many extensions [17]. Typically, optimal control requires defining a cost function or performance index, and the system dynamics becomes part of the constraints. Thus, determining the best control strategy is equivalent to minimizing the performance index, for example, the tracking error or control efforts, while following the constraints and boundary

conditions. Finding the optimal control input for a nonlinear system is not easy, but if the system can be linearized and operated around some specific points, the design tools for linear systems can be conveniently applied. Kim et al. compare the linearized optimal controller with feedback linearization and sliding mode control for a 3 DOF quadrotor in [39]. It is concluded that the optimal controller with gain scheduling tolerates even more model uncertainty than SMC. SMC produces best response but with the highest control effort, while the control effort of FBL lies in the middle of the controllers tested in simulation.

Based on our review of the control schemes, optimal control is chosen in this thesis. The reasons and some additional benefits of optimal control are summarized as follows. First of all, it offers standard and easy to implement algorithms for selecting the inner loop feedback gains automatically. Hence, the closed-loop stability and performance are guaranteed in theory. In contrast to the classical controller (like PID) design, in optimal control, all the feedback loops are closed simultaneously by solving standard matrix design equations. This is extremely useful for controlling a system with relatively large dimensions and coupled states like our airship. Secondly, as indicated in [39], extended optimal controller has advantages in terms of handling the system uncertainty. Since the airship itself is a time-variant system due to the helium leakage and battery drain, a model-sensitive strategy is not robust enough for our application. Thirdly, optimal control is energy-efficient: control efforts can be easily incorporated in the performance index and the thruster saturation can be alleviated by adjusting the parameters of the controller. This is in accord with the main advantage of using a lighter-than-air vehicle. At last, the desired strategy needs to be suitable for implementation in real time. Approaches that require large computational resources such as the nonlinear model predictive control [3], have therefore not been considered in this work.

1.3.3 State Estimation and Multi-Sensor Fusion

As a second goal to be accomplished in this thesis, the state estimation problem concerns with how to combine information from different sources (Vicon, IMU, LIDAR) with different sensor models and update rates. This overlaps with the topic of sensor fusion. Both topics have been studied for years in mobile robotics, and without exception in the context of UAVs. Since all systems and measurements have uncertainties, probabilistic methods such as Kalman filtering, are currently the mainstream choices [62].

A lot of state tracking and state estimation work on UAVs is done by combining GPS and IMU data because GPS is highly accurate compared with most on-board sensors, but not always reliable due to signal jamming or blocking by surrounding objects. On the other hand, an inertial measurement unit has a high bandwidth and reliable data updating but it is vulnerable to bias and drift over long term observation. In literature, the two measurements are used to complement each other in two ways: as loosely coupled or tightly coupled [44]. In tightly coupled systems, the raw data from GPS (namely pseudoranges) are directly used to correct IMU error growth, commonly by using some geometry model. On the other hand, in a loosely coupled integration, since the raw data is preprocessed by an internal Kalman filter first, the combination is more straightforward. In [14], to combine the data from a low-cost inertial navigation system (INS) and the GPS for a small-tethered blimp, Bijker et al. study a different architecture of Kalman filter. They find the trade-off between accuracy and processing requirements is achieved by having two small scaled extended Kalman filters (EKF) running in sequence. The first EKF estimates the airship attitude and passes the value to the second EKF, which estimates the velocity and position. It is worth noticing that their inertial measurement unit is somewhat similar to the one we have on-board our airship, containing a three-axes gyroscope, a three-axes accelerometer and a magnetometer. Their airship's attitude is determined by integrating the angular velocity with the correction from the magnetometer. In addition, the authors claim that the accelerometer measurements can also be fused in the attitude estimator since the airship's acceleration is trivial and the

acceleration vector is coincident with the gravity.

Another investigation of GPS/IMU fusion on a helicopter is found in [64]. First, the authors start with an EKF combining the gyroscope and accelerometer data with a kinematic model of the helicopter. GPS position and velocity measurements are then used to correct INS errors using the same EKF. Information such as helicopter's position, attitude, velocity as well as INS sensor biases are generated. Then, a sigma-point Kalman filter (SPKF) replaces the EKF and gains approximately 30% error reduction in both attitude and position estimation over the EKF results. A sensor latency compensation technique is also proposed to accurately fuse the lagged GPS data in [64].

Moving away from GPS/IMU integration, Kendoul et al. put forth a scheme called 3NKF (3 nested Kalman filters) to combine a low resolution wireless analog camera and a low cost IMU on a quadrotor [38]. Navigation and control algorithms are all deployed on-board in real-time except for the vision computation. The quadrotor motion is computed first with a Kalman filter using an optical flow algorithm and angular rate data; the output is then considered as measurements for the second Kalman filter in order to cancel the rotational component of optical flow; the translational component, nevertheless, is left to an EKF-based SFM (structure from motion) algorithm to recover.

Aa even more sophisticated system is found in [2] and [12], where a quadrotor equipped with a laser range finder, an IMU and a stereo-camera is built aiming at a completely autonomous air vehicle system that can take off, fly through a window, explore and map an unknown indoor environment, search for an object of interest, and transmit the video footage of the object location back to the ground station. To tackle complex tasks like this, a sensing and control hierarchy is essential. At the lowest level, a tight feedback loop is created based on the IMU to stabilize the quadrotor's pitch and roll. At the next level, state estimation of the position and velocity are obtained with an EKF combining the laser/visual odometry and the IMU outputs. The position control is thereby performed by a linear quadratic regulator (LQR) loop outside of the attitude feedback loop. In large environments,

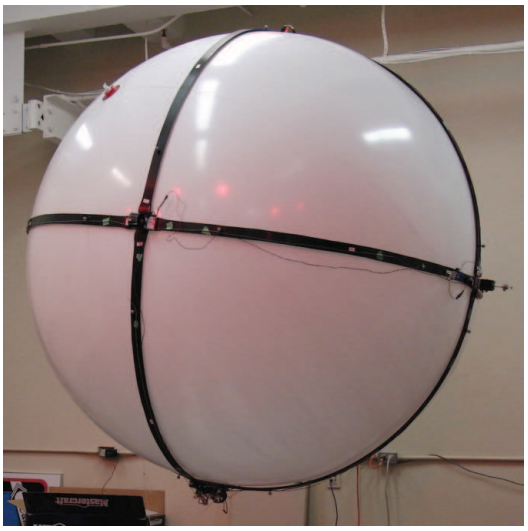
small errors made by the odometry algorithms will be propagated and result in inaccurate estimation. Therefore, at the third sensing level, a Simultaneous Localization and Mapping (SLAM) algorithm ensures globally consistent estimates of the vehicle pose while generating the map of environment. Moving higher in the control hierarchy, a planner makes waypoint trajectories through the known grid map maintained by the SLAM to reach the desired goal locations. Meanwhile, a mission planner coordinates the execution of a series of tasks, for example, autonomous takeoff, window entrance, target search, etc. To demonstrate fully autonomous operation of the vehicle in experiments, the quadrotor is tasked to explore and fly through office building hallways, with no human intervention.

There are many other state estimation methods not mentioned in this review, for example, particle filters which are more appropriate for global localization [63] and reinforcement-learning based state observation methods [40]. Since the work in this thesis focuses on well-established approaches that can be easily integrated with the proposed nonlinear controllers, we limit our investigation of state estimation solutions to Kalman filter based approaches.

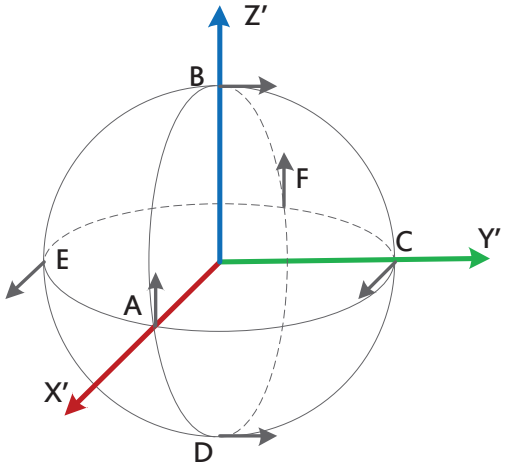
1.4 Experimental Equipment

1.4.1 Holonomic Airship

Fig. 1.1a presents the indoor airship discussed in this thesis. Nominally, it is a sphere full of helium with a 1.85 m (6 ft) diameter. It consists of a 0.05 mm (2 mil) thick plastic bladder and three carbon-fiber hoops with light-weight honey comb cores arranged orthogonal to each other to provide a rigid frame. At each of the six hoop intersection points is a thruster, a ducted fan, named from A to F. As will be shown in the next section, the fans generate different amounts of thrust depending on the direction of blade rotation. The direction of thruster is thereby defined as the direction of the air flow that generates the smaller force, as drawn in the airship frame $X'Y'Z'$ (Fig. 1.1b). In the meantime, the coordinate associated with the room in which the indoor flight is carried out is denoted as XYZ.



(a) Airship hovering in AML



(b) Airship frame and propeller arrangement, with thruster direction denoted by arrows

Figure 1.1 Indoor airship developed in AML

The key practice to fly the indoor airship is balancing, i.e., its center of gravity (CG) and center of buoyancy (CB) must be coincident. Since approximately 3.6 kg weight of components including the hull, are distributed on the surface of the airship, a slight imbalance will require considerable amount of power to compensate, not to mention the unknown impact the imbalance has on the dynamics model. Small aluminium weights are attached in the vicinity of the six thrusters to make fine adjustment of the balance, once all the other components are fixed. Finally, the neutral buoyancy of the airship is achieved by slightly inflating or deflating the helium inside. Even if perfectly balanced, it should be noted that, the airship is not an ideal system. The helium escapes through the bladder and eventually the airship starts to sink and lose its balance as the hull changes its size and shape, because the thickness of the bladder is not uniform and some surfaces areas are more elastic than others.

1.4.2 On-board Sensing, Power and Actuation

Two on-board sensors will be discussed in this thesis. The first one is a MicroStrain 3DM-GX1 inertia measurement unit (IMU), shown in Fig. 1.2a. This 75 g unit consists of three

angular rate gyros, three orthogonal accelerometers and three orthogonal magnetometers. It operates over the full 360 degrees of angular motion on all axes, providing orientation, angular velocities and translational accelerations with 16 bit A/D precision and 100 Hz sampling rate. Unfortunately, previous research [63] reveals that the acceleration information from the IMU is not usable because the airship accelerations are very low compared to the IMU's measurement range (±5g) and also because of the vibration noise caused by the thrusters. On the other hand, the orientation measurement can achieve a mean error under 7° compared to the ground-truth values from the off-board sensor. The orientation error is mainly attributed to the alignment error caused when the IMU is attached to the hoop, random walk of the gyroscope and the hard/soft magnetic interference resident in the environment. The first error can be compensated for by adding a bias to the measurements.

The other sensor on-board is a Hokuyo URG-04LX laser range scanner (LIDAR) shown in Fig. 1.2b. Being one of the tiniest commercial laser scanners available, it provides reliable distance information by emitting a laser beam and measuring the phase difference of the returned light. A single scan can span up to 240° with 0.36° angular resolution. The measurement accuracy is ±10 mm within 1 m and ±1% of the reading in the range of 1 m to 4 m, although the scanning rate is limited by the wireless communication between the sensor and the host computer (< 5 Hz for full scans [63]). A traditional LIDAR can only detect distance to objects, but after obtaining the protocol from Hokuyo, we are able to read the intensity of the returned light as well, which makes it possible to use retro-reflective material[1] as landmarks in navigation.

The power supply on-board the airship is comprised of two 2S2PL (two cells in series and two such series in parallel) lithium-polymer battery packs. One has a capacity of 4000 mAh and is used to power the thrusters and the motor driving circuit, while the other (480 mAh) is for the on-board sensors and communication. This is necessary because PWM control inevitably produces considerable ripples on the source, which can jeopardize the

[1]Retro-reflective material reflects light back to its source with a minimum scattering of light. In a LIDAR scan, a retro-reflective surface should have much higher intensity reading than a normal surface.

(a) 3DM-GX1 IMU **(b)** URG-04LX LIDAR **(c)** EDF-40 Ducted fan

Figure 1.2 On-board sensors and propellers

functionality and precision of the measurement components. The batteries are 8.4 VDC when fully charged, and are able to supply continuous current up to 16 C.[2]

There are six ducted fans (shown in Fig. 1.2c) mounted at the intersection of the hoops to provide two-way thrusts. Six H-bridge circuits drive these propellers as per the pulse-width modulation (PWM) signal (command ranging from 119 to 759 in our protocol) transmitted from the control station. The nominal maximum thrust of each fan is 73 g, but given the history of the system, all the thrusters have been calibrated in stationary tests with 8.4 VDC power supply, as shown in Fig. 1.3. Nonlinearities, like saturation and dead-zone, can be clearly observed in the diagram. According to these results, the maximum forward thrust is about 0.48 N while the maximum backward thrust is 0.28 N. A lookup table is programmed according to this diagram to map thrust and command signals to each other.

1.4.3 Off-board Sensing and Computing

The main off-board sensor installed in AML and used for airship control is the motion capture system from Vicon Motion System Inc. It consists of six M2 infra-red cameras

[2]For a lithium-polymer battery, C is the time it takes to discharge the battery in fractions of an hour. For instance 2 C discharges the battery in half an hour.

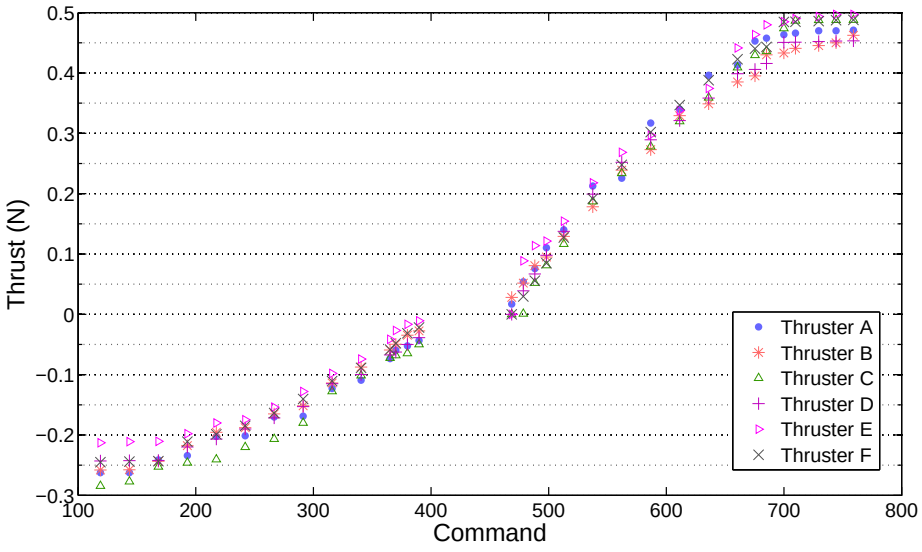


Figure 1.3 Thrust generated by propellers with respect to command received

and a V6 data station. The cameras are positioned in an arc on the ceiling so that infra-red light coming from the ring-shape emitters around the cameras fills the operating space of the airship. In order to capture the pose of the airship, a total of 24 retro-reflective markers are placed on its hoops, and the filter in front of the cameras ensures that only the light reflected by the markers is picked up. Every four-marker set surrounding each propeller has a unique pattern (different distances to each other) to form a rigid body used by the cameras. Each body carries the information on the airship's attitude and the position of the airship center. To obtain reliable and precise measurement, only the body observed by at least three cameras at the same time will be recognized and the body pose will be calculated based on epipolar geometry in stereo vision. In case multiple bodies are available, a special algorithm is prepared in the interpreting layer (see Section 1.4.4) to merge information for all.

As indicated earlier, a dedicated V6 station is used for 3D visual processing and body matching. Then, the data is transferred via a TCP connection to a control station[3] for post-processing. Since there is no computation unit on-board at present, the control station is also fully responsible for executing the control and estimation algorithms, as well as for

[3]Basic Configuration of the control station when this thesis is written is: CPU Intel Core2 6600; RAM 2 GB; Operating System Microsoft Windows XP

interfacing with sensors and thrusters. Also, as part of a user friendly robot system, the control station manages a graphic user interface (GUI).

1.4.4 Real-time Control Platform and System Interface

Quarc is Quanser's rapid prototyping and hardware-in-the-loop (HIL) platform for real time control. It works seamlessly with Matlab/Simulink to compile the Simulink model into real-time programs which can support various targets, such as the Quarc windows target, or QNX x86. The windows target allows all the code to be deployed on a personal computer under the Windows operating system (OS) rather than a real-time OS. This is realized by having a Quarc kernel running as a background service which gains the highest system priority. This kernel executes the compiled code from Simulink and manages all the I/O and communication. A subset of the Matlab library can be directly invoked in the user model. Meanwhile, Quarc also offers some flexibility to customize algorithm via the Matlab S function.

Since the control algorithm developed in the simulation program can be conveniently converted to executable code under Quarc, the key to successfully implementing the airship controller and estimator is the hardware interface. User models interact with hardware, for example, the data acquisition card or the serial port by using the blocks provided by the Quarc library. In the airship system, as shown in Fig. 1.4, the control station governs the data flow: a TCP connection with the Vicon V6 station, a serial connection with a Futaba six-channel RC control system to stream the PWM signal and two serial connections with bluetooth transceivers that communicate with the IMU and LIDAR respectively. Since Quarc only provides I/O level communication protocol, interpreting layers are programmed and compiled in C in order to translate sensor and control command values.

Based on Quarc and the interfaces connecting the user model and hardware, a PD controller has been developed as a starting point for controlling the indoor airship. The controller takes pose measurements from the Vicon system and angular velocity from the

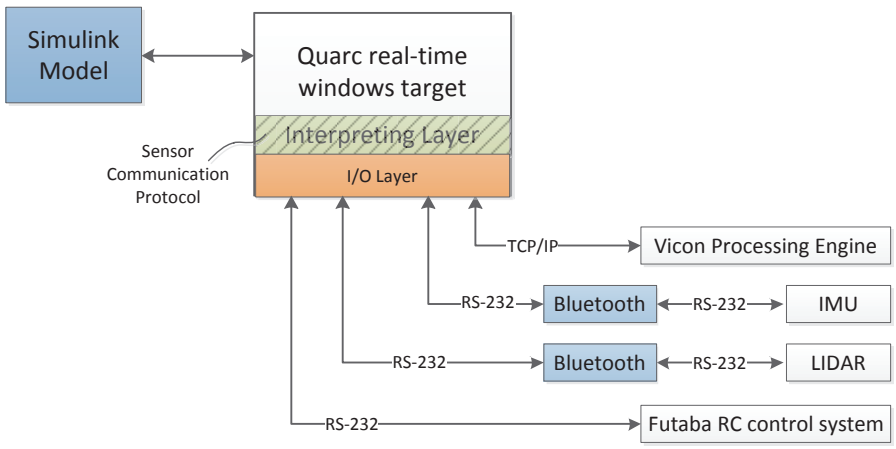


Figure 1.4 Quarc as core of airship control system and system interface

IMU. Without a dedicated sensor, the translational velocity is derived by finite differencing the position data and smoothing it with a finite impulse response (FIR) low-pass filter. No system dynamics have been accounted for in the PD controller design, and the PD parameters are selected and fine tuned in real flight.

Chapter 2

Airship Model and Its Linearization

2.1 State Definition

In order to attain full pose controllability, the state definition should at least contain airship orientation, position of the geometric center, angular velocity and translational velocity. The state definition has been changed from the one employed previously for design of the PD controller, specifically with regard to the orientation representation. There are four most commonly used options: an orientation matrix may be the most straightforward way, but it augments the state by 9 elements which will increase the computational requirement; the Euler angle representation is widely accepted in aerial vehicle and underwater vehicle control, but it suffers from a singularity problem known as Gimbal lock which should be avoided in our six-degree-of-freedom airship [33]; Euler angle/axis is our previous paradigm in developing the PD controller (defined by vector $[R_x \quad R_y \quad R_z]^T$), since it is the only raw orientation representation available from the Vicon system. However, a cumbersome nonlinear projection involving square root and trigonometry is needed in order to construct the orientation matrix $\mathbf{R}$, as stated in Eqs. (2.1–2.3). This can result in difficulty in the linearization process.

$$\theta = \sqrt{R_x^2 + R_y^2 + R_z^2} \tag{2.1}$$

$$\mathbf{e} = \frac{1}{\sqrt{R_x^2 + R_y^2 + R_z^2}} \begin{bmatrix} R_x & R_y & R_z \end{bmatrix}^T \tag{2.2}$$

$$\mathbf{R} = \mathbf{I_3} \cos\theta + \mathbf{I_3}(1 - \cos\theta)\mathbf{e}\mathbf{e}^T + \mathbf{e}^\times \times \sin\theta \tag{2.3}$$

where $\mathbf{I_3}$ is a 3-by-3 identity matrix and

$$\mathbf{e}^\times = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{2.4}$$

In the end, the quaternion defined as $\mathbf{q} = [q_0 \quad q_1 \quad q_2 \quad q_3]^T$ becomes our final solution. By adding one dimension to the Euler axis and performing the trigonometry transformation as in Eqs. (2.5–2.6), one can readily obtain the orientation matrix $\mathbf{R}$ which takes a vector in the rotating frame to the base frame and is given in Eq. (2.7) [7].

$$q_0 = \cos(\theta/2) \tag{2.5}$$

$$\begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T = \mathbf{e}\sin(\theta/2) \tag{2.6}$$

$$\mathbf{R} = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_2q_0 \\ 2q_1q_2 + 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 - 2q_1q_0 \\ 2q_1q_3 - 2q_2q_0 & 2q_2q_3 + 2q_1q_0 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \tag{2.7}$$

Meanwhile, for constructing the transformation matrix which connects the velocities and the derivative of pose, one needs to associate the derivative of orientation representation and the angular velocity. In that respect, the relationship between quaternion derivative and angular velocity is available and it is bilinear [27][18], as given by:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{Q}_\omega \boldsymbol{\omega} \tag{2.8}$$

$$\mathbf{Q}_\omega = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix} \tag{2.9}$$

Another decision to be made with respect to the state definition is in which frame the position and velocity vectors should be written. First, the sensors for pose observation, either Vicon system or the on-board laser range finder, are generating data in the global frame either directly or after frame transformation. Also one of the main goals for the design of the airship control system is to attain global pose controllability. Thus, only the absolute position and attitude are provided as reference signals. Of course, the intuitive way is to make every element in the state definition consistent, meaning that all vectors are written in the global frame as was previously implemented for the PD controller. However, the thruster and IMU inputs are inherently attached to the airship frame. Therefore, a computationally demanding inertia matrix inverse problem is inevitable down the road. The Jacobian linearization is impractical based on this state definition.

In view of the above considerations, the best alternative is to express the orientation and translation vectors in the global frame, while leaving all the velocity quantities in the airship or body fixed frame. By doing so, we will shift the frame transformation to the orientation and translation parts of the state matrix which results in lighter computational load (as detailed in the next section.) The state definition to be used to formulate the state space model is thus given as follows:

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & p_X & p_Y & p_Z & \omega_x & \omega_y & \omega_z & v_x & v_y & v_z \end{bmatrix}^T \\ &= \begin{bmatrix} \mathbf{q}^T & \mathbf{p}^T & \boldsymbol{\omega}^T & \mathbf{v}^T \end{bmatrix}^T \end{aligned} \tag{2.10}$$

where $\mathbf{p} = [p_X \quad p_Y \quad p_Z]^T$ contains the coordinates of the airship center in the global frame; $\boldsymbol{\omega} = [\omega_x \quad \omega_y \quad \omega_z]^T$ is the angular velocity in the airship frame; $\mathbf{v} = [v_x \quad v_y \quad v_z]^T$ is the translational velocity in the airship frame. It is worth mentioning here, as a prerequisite

for the controller and estimator design discussed in the following chapters, that there is a constraint on the norm of the quaternion:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{2.11}$$

Differentiating the above with time and solving for $\dot{q}_0$ by substituting the derivatives $\dot{q}_1$, $\dot{q}_2$ and $\dot{q}_3$ obtained from Eqs. (2.8–2.9) gives:

$$\begin{aligned}
\dot{q}_0 &= -\frac{1}{q_0}(q_1\dot{q}_1 + q_2\dot{q}_2 + q_3\dot{q}_3) \\
&= -\frac{1}{q_0}\left[q_1(\frac{1}{2}q_0\omega_x + \frac{1}{2}q_3\omega_y - \frac{1}{2}q_2\omega_z) + q_2(-\frac{1}{2}q_3\omega_x + \frac{1}{2}q_0\omega_y + \frac{1}{2}q_1\omega_z)\right. \\
&\quad \left. + q_3(\frac{1}{2}q_2\omega_x - \frac{1}{2}q_1\omega_y + \frac{1}{2}q_0\omega_z)\right] \\
&= -\frac{1}{2}(q_1\omega_x + q_2\omega_y + q_3\omega_z)
\end{aligned} \tag{2.12}$$

which gives the first row of $\mathbf{Q}_\omega$. Therefore, the state matrix formulated with $\mathbf{Q}_\omega$ is not full rank, which means that the system is overdetermined. Therefore, it easily becomes uncontrollable in a few simulation steps if solution of an algebraic Riccati equation (ARE) is required by the controller (the ARE will be introduced in §3.1). To remedy this problem while keeping the quaternion representation, we redefine the state by dropping the redundant q_0 so that the state definition becomes:

$$\mathbf{x} = \begin{bmatrix} q_1 & q_2 & q_3 & \mathbf{p}^T & \boldsymbol{\omega}^T & \mathbf{v}^T \end{bmatrix}^T \tag{2.13}$$

This reduces the dimension of the state matrix to 12×12 and makes it full rank. However, q_0 is still required and it can be recovered either directly from the state estimation, or by using $q_0 = \pm\sqrt{1 - q_1^2 - q_2^2 - q_3^2}$. For the latter case, however, determining the sign of q_0 requires a cumbersome case analysis of the angular velocity and the quaternion at the previous time step. If not dealt with carefully, the dynamics model and the state estimator may suffer from a singular attitude, for example, when $\mathbf{q} = [\cos(\pi/2) \quad 0 \quad \sin(\pi/2) \quad 0]^T$, and the simulation will diverge. In the end, the 12 dimensional state definition will only apply to the ARE related design (§3.3 and §3.4). In the rest of this thesis, we will use the 13 dimensional state

definition as given in Eq. (2.10).

2.2 Airship Model

The airship dynamics equations can be found in previous reports and publications [51][32][43]. The airship's motion can be effectively regarded as a free body movement subject to the aerodynamic effects, and the thruster forces.

The model presented here is based on the following assumptions:

1. The airship has been perfectly balanced, i.e., the center of mass and the center of buoyancy are coincident at the geometric center of the sphere. As well, the airship is neutrally buoyant so that the buoyancy force cancels the gravity force.
2. Any unpredictable disturbance, for example the indoor airflow, has been ignored.
3. The system is time-invariant over the flight period, i.e., all events that may change the parameters of the system, for example, helium leakage, battery drain, are ignored.

The system dynamics can be formulated in the airship frame as:

$$(m_a + m_{add})(\dot{\mathbf{v}} + \boldsymbol{\omega} \times \mathbf{v}) = \sum_{i=1}^{6} \mathbf{f}_{Ti} + \mathbf{f}_D + \mathbf{f}_L \tag{2.14}$$

$$\mathbf{I}_a \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}_a \boldsymbol{\omega}) = \sum_{i=1}^{6} \mathbf{r}_i \times \mathbf{f}_{Ti} + \mathbf{M}_D \tag{2.15}$$

where $\mathbf{v}$ and $\boldsymbol{\omega}$ are as defined in §2.1; $\mathbf{I}_a$ is the inertia matrix of the airship; $\mathbf{r}_i$ denotes the location of the thrusters relative to the airship center; $\mathbf{f}_{Ti}$ is the thruster inputs; $\mathbf{f}_D$ is the translational drag on an ideal sphere computed as [20]:

$$\mathbf{f}_D = -\frac{\pi}{2} \rho_{air} C_D r_a^2 \|\dot{\mathbf{p}}\| \dot{\mathbf{p}} \tag{2.16}$$

The force $\mathbf{f}_L$ is the aerodynamic lift (Magus force) defined as [26]:

$$\mathbf{f}_L = \frac{\pi}{2} \rho_{air} C_L r_a^2 \|\dot{\mathbf{p}}\|^2 \frac{\boldsymbol{\omega} \times \dot{\mathbf{p}}}{\|\boldsymbol{\omega} \times \dot{\mathbf{p}}\|} \tag{2.17}$$

and $\mathbf{M}_D$ is the rotational drag [55]:

$$\mathbf{M}_D = -\frac{\pi}{2}\rho_{air} C_R r_a^5 \|\boldsymbol{\omega}\| \boldsymbol{\omega} \tag{2.18}$$

As the airship changes its velocity, extra effort is required to accelerate some volume of the surrounding air, so that the airship inertia needs to be corrected. This effect is called added mass or virtual mass. For a spherical particle submerged in an inviscid, incompressible fluid, it can be shown that [25]

$$m_p \frac{dv_p}{dt} = \sum F + \frac{\rho_c V_p}{2}\left(\frac{du_c}{dt} - \frac{dv_p}{dt}\right) \tag{2.19}$$

where v_p is the velocity of the particle, m_p is its mass while V_p is its volume. ρ_c is the density of the fluid, u_c is the fluid flow velocity and F represents the net external force. Since the airship moves slowly, the surrounding air can be assumed to be incompressible and non-viscous. In the indoor test environment, the air velocity can be assumed negligible. Therefore, Eq. (2.19) can be rearranged as

$$\left(m_p + \frac{\rho_c V_p}{2}\right)\frac{dv_p}{dt} = \sum F \tag{2.20}$$

which clearly shows the additional virtual mass to the particle mass. In our scenario, the fluid is air with density ρ_a and V_a and m_a are the volume and mass of the airship respectively and:

$$m_{add} = \frac{1}{2}\rho_a V_a = \frac{1}{2} m_a \tag{2.21}$$

The system presented in Eqs.(2.14–2.15) is highly nonlinear, mainly due to the aerodynamics forces. It is natural to first examine the impact of aerodynamics on the airship before proceeding with linearisation, since the aerodynamic coefficients C_D, C_R, C_L are all associated with the Reynolds number, which depends on the velocities of the airship. The empirical formulas to compute the aerodynamic coefficients have been studied extensively [52][47][26][20][55]. Generally, these are highly nonlinear and sometimes discontinuous functions, which means it is difficult to establish closed-form bounds on these terms. Thereby the velocity space of the airship (relatively a low Reynolds number space) is meshed, and

scanned to examine the magnitudes of the aerodynamic terms. The velocity boundary is defined by $\omega_{\max} = 1$ rad/s for rotational velocity $v_{\max} = 0.5$ m/s and for translational velocity. They are the peak values observed in the flight records to date. To obtain a conservative boundary, we assume the translational and angular velocities to be orthogonal.

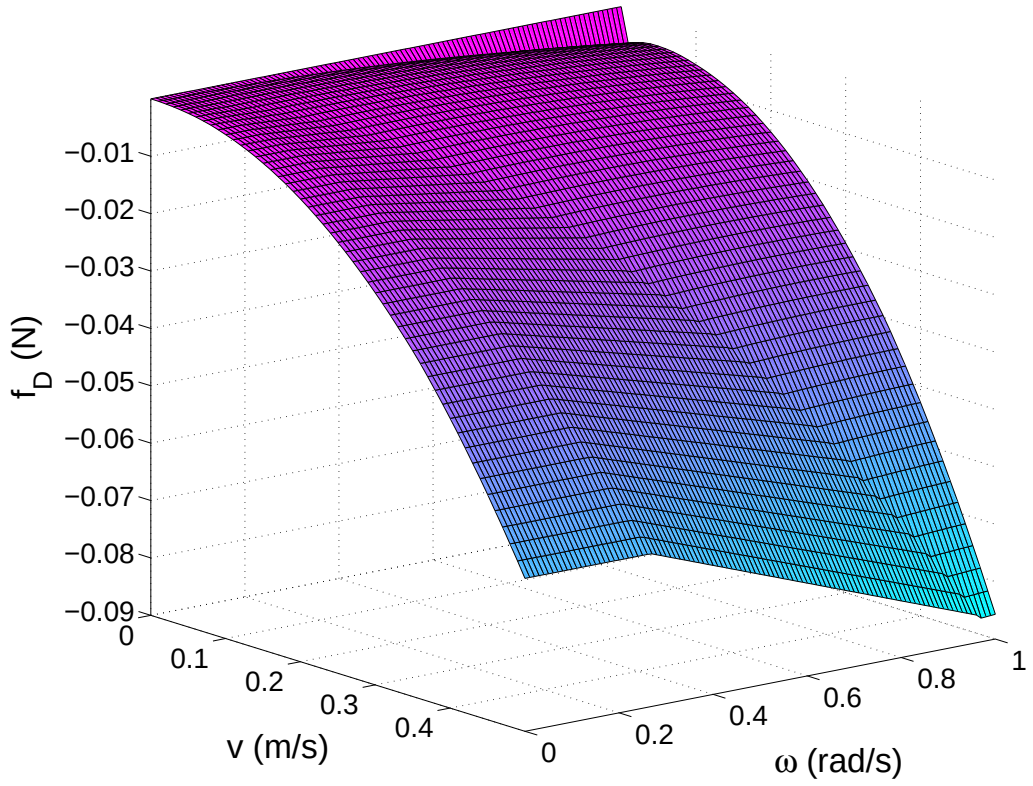


Figure 2.1 Translational drag with respect to the velocities

The magnitudes of the three aerodynamic terms, $\mathbf{f}_D$, $\mathbf{M}_D$ and $\mathbf{f}_L$ are shown in Fig. 2.1-2.3, respectively. These should be compared to the thruster's maximum force of 0.48 N and the thruster's maximum moment of 0.45 Nm. Compared with the thrust exerted on the airship, among the aforementioned aerodynamic effects, the rotational drag is negligible in the full range of velocities(Fig. 2.2). The translational drag (Fig. 2.1) and the Magnus lift (Fig. 2.3) may need to be taken into account when both $\boldsymbol{\omega}$ and $\mathbf{v}$ are high, which rarely happens. Therefore, the aerodynamic terms can be safely dropped from the system equations without significantly compromising on their accuracy. The airship dynamics can thus be written in

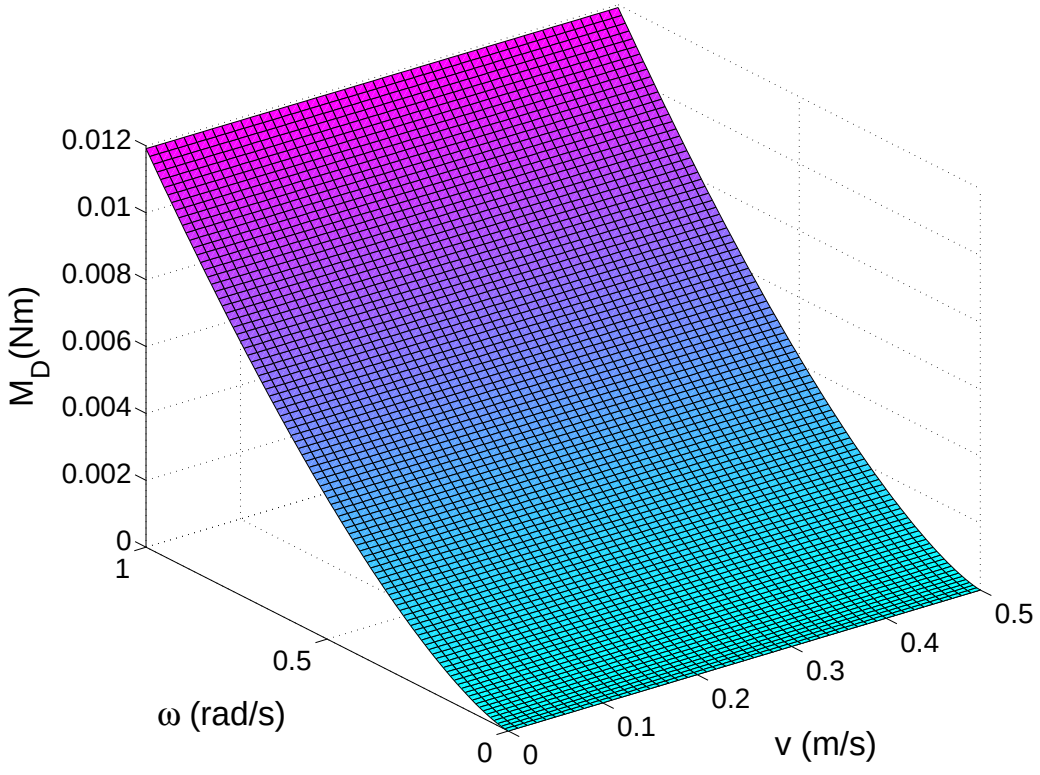


Figure 2.2 Rotational drag with respect to the velocities

the general form:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.22}$$

where $\mathbf{u} = [u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6]^T$ is the system input, specifically, the thrusts of the six propellers $\mathbf{f}_T \in \mathbb{R}^{3\times 6}$ can be combined as:

$$\mathbf{f}_T = \begin{bmatrix} 0 & 0 & u_3 & 0 & u_5 & 0 \\ 0 & u_2 & 0 & u_4 & 0 & 0 \\ u_1 & 0 & 0 & 0 & 0 & u_6 \end{bmatrix} \tag{2.23}$$

Hence, the system dynamics are:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{p}} \\ \dot{\boldsymbol{\omega}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\mathbf{Q}_\omega \mathbf{R}\boldsymbol{\omega} \\ \mathbf{R}\mathbf{v} \\ \mathbf{I}_a^{-1}(\sum \mathbf{r}_i \times \mathbf{f}_{Ti} - \boldsymbol{\omega} \times (\mathbf{I}_a \boldsymbol{\omega})) \\ \frac{2}{3m_a}\sum \mathbf{f}_{Ti} - \boldsymbol{\omega} \times \mathbf{v} \end{bmatrix} \tag{2.24}$$

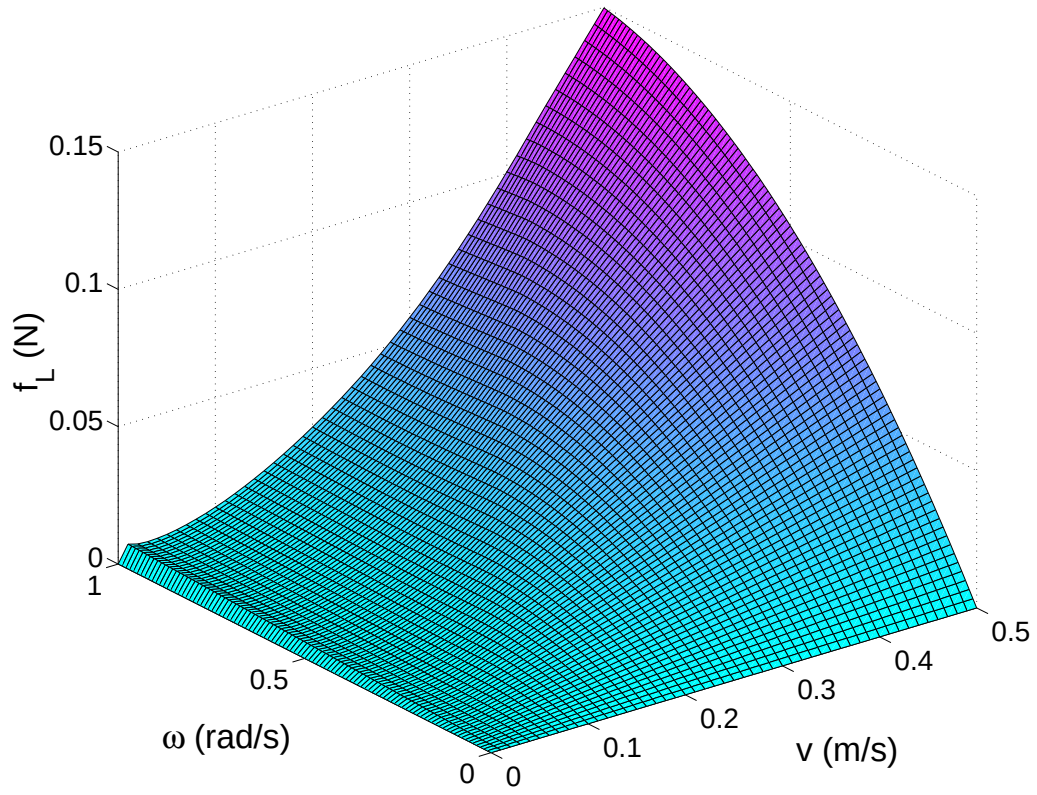
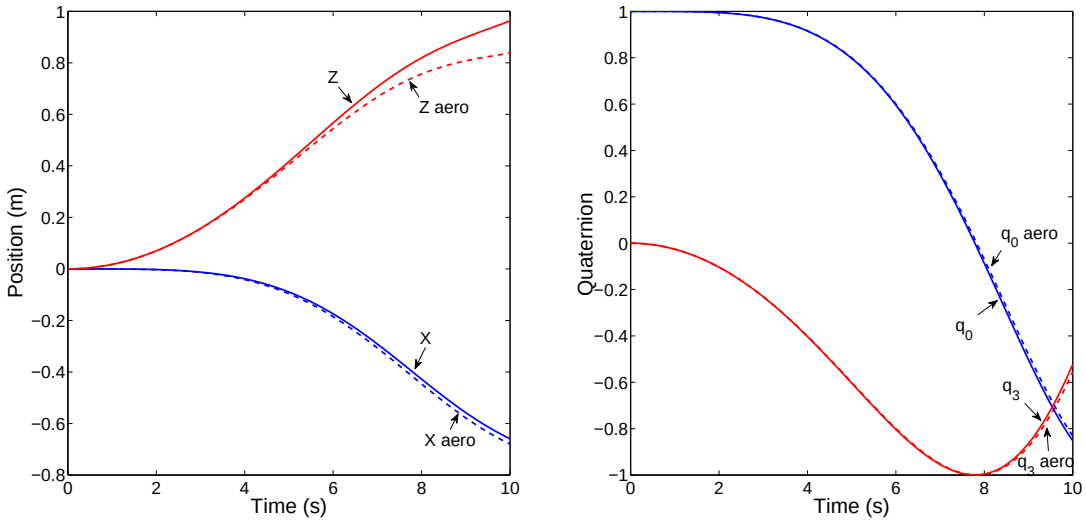


Figure 2.3 Aerodynamic lift with respect to the velocities

where $\mathbf{f}_{Ti}$ is the ith column of $\mathbf{f}_T$. The above equations form the basis of the linearized model and controller design presented in Chapter 3 of this thesis.

In order to examine the argument that the aerodynamics can be ignored for our indoor application, the aforementioned nonlinear model with and without the aerodynamic terms has been used in simulation. Starting from initial attitude $\mathbf{q} = [1 \quad 0 \quad 0 \quad 0]^T$, a constant thrust (0.2 N) is applied on propeller A (see Fig. 1.1b) for 10 s, such that the airship will spin along axis Y and translate in the XZ plane. Thus the three aerodynamic effects will all have an impact on this maneuver. We compare the results obtained without aerodynamic terms to those obtained with aerodynamic terms. The corresponding results are plotted in Fig. 2.4. With regard to translational motion, the major difference is in the translation along the Z axis, which achieves a higher velocity earlier than the X axis, thus affected more by the translational drag. On the other hand, no significant difference is observed in the rotational motion. It needs to be pointed out that, in real flights, the control input is rarely

left open-loop and lasts sufficiently long to generate substantial translation of the airship like the simulation. The aerodynamic differences, along with other unmodeled effects such as imbalance and battery drain, will play the role of disturbances after closing the control loop.



(a) Position **(b)** Attitude

Figure 2.4 Position and attitude difference of the airship model with and without aerodynamics. The axes without significant change are ignored.

2.3 Jacobian Linearisation

First we examine the equilibrium family[1] of Eq. (2.22). defined by:

$$\mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) = \mathbf{0} \tag{2.25}$$

Since the airship is self-balanced in any configuration in the absence of control, one set of equilibrium solutions can be readily proven to be

$$\boldsymbol{\omega}_0 = \mathbf{0}, \quad \mathbf{v}_0 = \mathbf{0}, \quad \mathbf{u}_0 = \mathbf{0} \qquad \forall \mathbf{q}_0, \mathbf{p}_0 \tag{2.26}$$

i.e., $\mathbf{x}_0$ can represent any arbitrary pose as long as the velocities and thrusts are null. For cases other than those obtained by Eq. (2.26), if we expand $f(\mathbf{x}(t), \mathbf{u}(t))$ at state $\mathbf{x_0}$ given at

[1] By equilibrium family, we mean the set of all equilibirum states and inputs derived from a set of linearizations (rather than just a single linearization) [41].

time t using Taylor series and retain linear terms only to obtain

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t)) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_0 \delta\mathbf{x}(t) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_0 \delta\mathbf{u}(t) \tag{2.27}$$

Defining the state perturbation vector and the control correction vector as:

$$\delta\mathbf{x}(t) \triangleq \mathbf{x}(t) - \mathbf{x_0}(t) \tag{2.28}$$

$$\delta\mathbf{u}(t) \triangleq \mathbf{u}(t) - \mathbf{u_0}(t) \tag{2.29}$$

and recalling that

$$\mathbf{f}(\mathbf{x}_0(t), \mathbf{u}_0(t)) = \mathbf{0} \tag{2.30}$$

we can readily derive the linearized system

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}_0(t)\delta\mathbf{x}(t) + \mathbf{B}_0(t)\delta\mathbf{u}(t) \tag{2.31}$$

In the preceding, $\mathbf{A}_0$ and $\mathbf{B}_0$ are the Jacobian matrices computed at $\mathbf{x_0}(t)$

$$\mathbf{A}_0(t) \triangleq \left.\frac{\partial f}{\partial \mathbf{x}}\right|_0 = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}_0(t) \\ \mathbf{u}_0(t)}} \tag{2.32}$$

$$\mathbf{B}_0(t) \triangleq \left.\frac{\partial f}{\partial \mathbf{u}}\right|_0 = \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}_0(t) \\ \mathbf{u}_0(t)}} \tag{2.33}$$

where $\mathbf{A}_0$ is a 13×13 matrix for our system. For convenience of presentation, it is partitioned into four sub-matrices.

$$\mathbf{A}_0 = \begin{bmatrix} \mathbf{A}_q \\ \mathbf{A}_p \\ \mathbf{A}_\omega \\ \mathbf{A}_v \end{bmatrix} \tag{2.34}$$

where $\mathbf{A}_q$ is a 4×13 matrix, $\mathbf{A}_p$, $\mathbf{A}_\omega$ and $\mathbf{A}_v$ are 3×13 matrices with the corresponding explicit forms given in Appendix A. Similarly, $\mathbf{B}_0$ can be written as

$$\mathbf{B}_0 = \begin{bmatrix} \mathbf{0}_{7\times6} \\ \mathbf{b}_{t1} \\ \mathbf{b}_{t2} \\ \mathbf{b}_{t3} \\ \mathbf{B}_f \end{bmatrix} \tag{2.35}$$

where $\mathbf{b}_{t1}$, $\mathbf{b}_{t2}$ and $\mathbf{b}_{t3}$ are 1×6 matrix and $\mathbf{B}_f$ is a 3×6 matrix given in Appendix A. All the linearization work is done by using the Symbolic Math Toolbox in Matlab.

The linearized model is validated in simulation by comparing its results to those of the nonlinear model. Since the system nonlinearity mainly arises from the orientation transformation, the simulated maneuver is defined by producing a small amount of rotation near the equilibrium point. The initial attitude of the airship is chosen to be $\mathbf{q} = [0.616 \quad 0.455 \quad 0.455 \quad 0.455]^T$, and a sinusoidal couple (0.2 N in each propeller, 5 s in period) is applied at propellers A and F. As shown in Fig. 2.5, the differences between the linear and nonlinear model results are insignificant along the three axes. Therefore, we can use the linearized model to design the controller at a specific pose.

Having derived and validated the linearized model, the airship control problem can be stated as follows: given a set point or desired trajectory to be followed by the airship, design a controller to achieve the desired motion while rejecting perturbations.

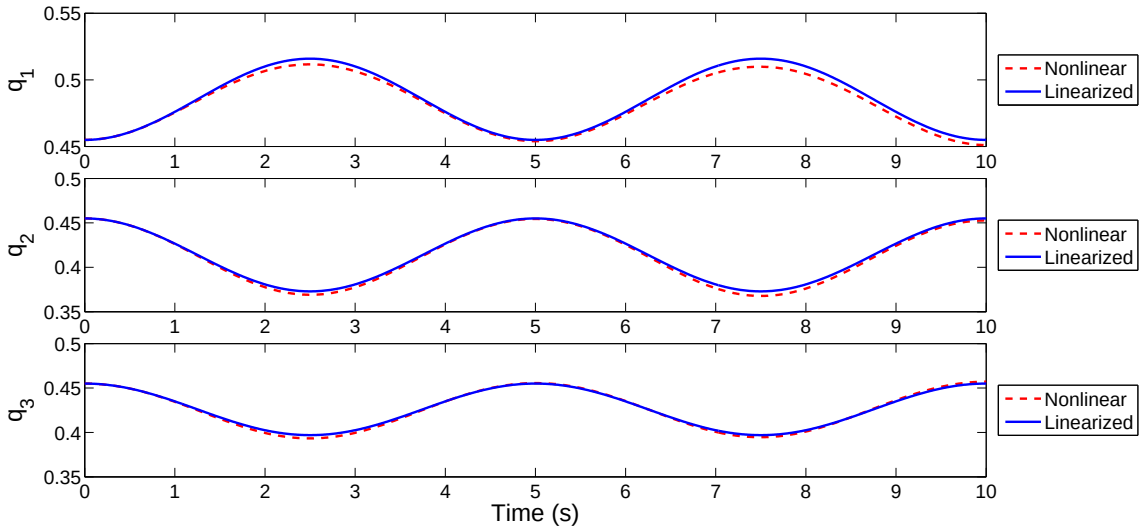


Figure 2.5 Comparison of the linearized model and the nonlinear model

Chapter 3

Optimal Control of Airship: Theory and Simulation

3.1 Systematic View of Deterministic Optimal Control

In the classical controller design (such as the PD controller), the control loop is closed one variable at a time by directly selecting the feedback gains, mostly relying on experience of the designer and careful tuning on-site. Unfortunately, for a system containing 13 states like our airship, neither stability nor robustness of the overall system can be guaranteed in this way. For example, improper feedback loop design of the angular velocity control can compromise the attitude stability of the airship. On the other hand, modern optimal control offers standard algorithms for selecting the inner loop feedback gains automatically. Hence the closed-loop stability and performance are guaranteed in theory. In contrast to the classical controller, in optimal control, all the feedback loops are closed simultaneously by solving standard matrix design equations. A general formulation of the optimal control

problem is given by equations as follows:

$$\min \qquad J(t_0) = \phi\left(\mathbf{x}\left(T\right), T\right) + \int_{t_0}^{T} l(\mathbf{x}, \mathbf{u}, t)dt \tag{3.1}$$

s.t.

$$\textbf{Dynamics constraints:} \qquad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{3.2}$$

$$\textbf{Final state constraints:} \qquad \boldsymbol{\Psi}(\mathbf{x}(T), T) = \mathbf{0} \tag{3.3}$$

where t_0 and T are the initial and final times for the task, ϕ is the isolated point of the performance index J evaluated at T while l is accumulated during $t \in [t_0, T]$. The ultimate goal of optimal control is to minimize the performance index J under the two constraints stated in Eqs. (3.2-3.3). There is no design algorithm available to solve this general nonlinear optimal control problem at the present. However, for systems which can be described by linear dynamics constraints, as for example, with the linearized model of the airship, the system constraint Eq. (3.2) can be rewritten as

$$\dot{\mathbf{x}} = \mathbf{A}_0(t)\mathbf{x} + \mathbf{B}_0(t)\mathbf{u} \qquad t \in [t_0, T] \tag{3.4}$$

Choosing the control weighting matrix $\mathbf{R}_L$, the state weighting matrix $\mathbf{Q}_L$, the final state weighting matrix $\mathbf{F}_L$, and zeroing the initial state, the goal of the linear optimal control then becomes minimizing the quadratic performance index $J(\mathbf{x}(t_0), \mathbf{u}(\cdot), t_0)$ which depends on the state $\mathbf{x}$, the initial time t_0 and the unknown control $\mathbf{u}$ over $[t_0, T]$:

$$J(\mathbf{x}(t_0), \mathbf{u}(\cdot), t_0) = \mathbf{x}^T(T)\mathbf{F}_L\mathbf{x}(T) + \int_{t_0}^{T} \left[\mathbf{x}^T(t)\mathbf{Q}_L\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}_L\mathbf{u}(t)\right]dt \tag{3.5}$$

Comparing with the nonlinear problem formulation in Eqs. (3.1-3.3), we have

$$\phi\left(\mathbf{x}\left(T\right), T\right) = \mathbf{x}^T(T)\mathbf{F}_L\mathbf{x}(T)$$

$$l(\mathbf{x}, \mathbf{u}, t) = \mathbf{x}^T(t)\mathbf{Q}_L\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}_L\mathbf{u}(t)$$

The additional condition on the weighting matrices is the positive definitive constraint, which is

$$\mathbf{F}_L = \mathbf{F}_L^T \geq 0, \quad \mathbf{Q}_L = \mathbf{Q}_L^T \geq 0, \quad \mathbf{R}_L = \mathbf{R}_L^T > 0$$

Let J^* represent the optimal (minimal) value of the performance index:

$$J^*(\mathbf{x}(t), t) = \min_{\mathbf{u}[t,T]} J(\mathbf{x}(t), \mathbf{u}(\cdot), t_0) \tag{3.6}$$

By the Principle of Optimality [5], the last part of an optimal trajectory is also optimal, so that,

$$\forall t_1 \in [t, T] \quad J^*(\mathbf{x}(t), t) = \min_{u[t,T]} \left[\int_t^{t_1} l(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau + J^*(\mathbf{x}(t_1), t_1) \right] \tag{3.7}$$

Expanding the above equation by applying Taylors theorem and then taking the time derivative, we obtain:

$$\frac{\partial J^*}{\partial t}(\mathbf{x}(t), t) = -\min_{u[t,T]} \left\{ l\left(\mathbf{x}(t), \mathbf{u}(t), t\right) + \left[\frac{\partial J^*}{\partial \mathbf{x}}(\mathbf{x}(t), t) \right]^T \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \right\} \tag{3.8}$$

For a linearized system like the airship presented here,

$$\mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) = \mathbf{A}_0(t)\mathbf{x}(t) + \mathbf{B}_0(t)\mathbf{u}(t)$$

Equation (3.8) represents one statement of the Hamilton-Jacobi-Bellman (HJB) equation. It can be proven [4] that $J^*(\mathbf{x}(t), t)$ can be rewritten in the following form where $\mathbf{K}$ is symmetric:

$$J^*(\mathbf{x}(t), t) = \mathbf{x}^T(t)\mathbf{K}(t)\mathbf{x}(t) \tag{3.9}$$

With the above, Eq. (3.8) in this quadratic case becomes the statement of the regulator problem:

$$\frac{\partial J^*}{\partial t} = \mathbf{x}^T \dot{\mathbf{K}} \mathbf{x}^T = -\min_{\mathbf{u}} \left[\mathbf{u}^T \mathbf{R}_L \mathbf{u} + \mathbf{x}^T \mathbf{Q}_L \mathbf{x} + 2\mathbf{x}^T \mathbf{K} \mathbf{A}_0 \mathbf{x} + 2\mathbf{x}^T \mathbf{K} \mathbf{B}_0 \mathbf{u} \right] \tag{3.10}$$

To find the minimum on the right hand side of Eq. (3.10), we need to isolate a quadratic expression in $\mathbf{u}$ and this can be done by completing the square as follows:

$$\begin{aligned} &\mathbf{u}^T \mathbf{R}_L \mathbf{u} + \mathbf{x}^T \mathbf{Q}_L \mathbf{x} + 2\mathbf{x}^T \mathbf{K} \mathbf{A}_0 \mathbf{x} + 2\mathbf{x}^T \mathbf{K} \mathbf{B}_0 \mathbf{u} \\ =&\left(\mathbf{u} + \mathbf{R}_L^{-1} \mathbf{B}_0^T \mathbf{K} \mathbf{x}\right)^T \mathbf{R}_L \left(\mathbf{u} + \mathbf{R}_L^{-1} \mathbf{B}_0^T \mathbf{K} \mathbf{x}\right) \\ &+ \mathbf{x}^T (\mathbf{Q}_L - \mathbf{K} \mathbf{B}_0 \mathbf{R}_L^{-1} \mathbf{B}_0^T \mathbf{K} + \mathbf{K} \mathbf{A}_0 + \mathbf{A}_0^T \mathbf{K}) \mathbf{x} \end{aligned} \tag{3.11}$$

Since the matrix $\mathbf{R}_L$ is positive definite, it follows that Eq. (3.8) is minimized by setting

$$\mathbf{u} + \mathbf{R}_L^{-1}\mathbf{B}_0^T\mathbf{K}\mathbf{x} = 0 \tag{3.12}$$

Noting that Eq. (3.10) holds for all $\mathbf{x}$ and both sides of it are symmetric, we obtain a differential equation for $\mathbf{K}(t)$:

$$-\dot{\mathbf{K}}(t) = \mathbf{Q}_L(t) - \mathbf{K}(t)\mathbf{B}_0(t)\mathbf{R}_L^{-1}\mathbf{B}_0^T(t)\mathbf{K}(t) + \mathbf{K}(t)\mathbf{A}_0(t) + \mathbf{A}_0^T(t)\mathbf{K}(t) \tag{3.13}$$

Recalling the boundary condition of the Hamilton-Jacobi equation,

$$J^*(\mathbf{x}(T), T) = \mathbf{x}^T(T)\mathbf{F}_L\mathbf{x}(T) \tag{3.14}$$

combined with Eq. (3.9), we have the boundary condition for $\mathbf{K}(t)$:

$$\mathbf{K}(T) = \mathbf{F}_L \tag{3.15}$$

As a closure for the above derivation, in the linear optimal control, the feedback gain $\mathbf{G}(t)$ is determined by:

$$\mathbf{u}(t) = -\mathbf{G}(t)\mathbf{x}(t) = -\mathbf{R}_L^{-1}\mathbf{B}_0^T(t)\mathbf{K}(t)\mathbf{x}(t) \tag{3.16}$$

where $\mathbf{K}$ is obtained from the differential Riccati equation (3.13) and the boundary condition (3.15).

Therefore, we discussed the continuous form of the linear quadratic regulator. A special case of the continuous quadratic method is the infinite horizon regulator. For a time-invariant system, $\mathbf{A}_0$ and $\mathbf{B}_0$ in Eq. (3.4) are constant matrices, and if T approaches infinity in the performance index defined in Eq. (3.5), the Riccati equation will have a steady-state solution where $\dot{\mathbf{K}} = \mathbf{0}$. In this case, the design problem reduces to solving an algebraic Riccati equation (ARE) given by:

$$\mathbf{K}\mathbf{A}_0 + \mathbf{A}_0^T\mathbf{K} + \mathbf{Q}_L - \mathbf{K}\mathbf{B}_0\mathbf{R}_L^{-1}\mathbf{B}_0^T\mathbf{K} = \mathbf{0} \tag{3.17}$$

In the optimal controller synthesis, the weighting matrices $\mathbf{Q}_L$, $\mathbf{R}_L$ and $\mathbf{F}_L$ are the primary design parameters. Usually they are selected by the designer on the basis of engineering

experience, physical characteristics of the system and simulation studies. There is no universal agreement on how they are to be selected for any given application [11]. However, there are some general principles that can be followed, as stated below:

1. In order to alleviate the dependency among components of the state vector and those of the control input, $\mathbf{Q}_L$ and $\mathbf{R}_L$ are selected to be diagonal. In this manner, one can penalize the components in $\mathbf{x}$ and $\mathbf{u}$ individually.
2. The larger $\|\mathbf{F}_L\|$, the larger is the gain matrix $\mathbf{G}(t)$ for values of time near the terminal time. In our application, in the absence of other specifications, $\mathbf{F}_L$ is typically assigned to be $\mathbf{0}$[1] of airship control because the control should be shut down as the airship approaches the end of the task.
3. The larger $\|\mathbf{Q}_L\|$, the larger is the gain matrix $\mathbf{G}(t)$ and hence the shorter is the time for the state perturbations to be reduced. The ultimate performance can be adjusted by the diagonal elements in $\mathbf{Q}_L$ according to the flight results. For evaluating the controller in simulation, $\mathbf{Q}_L$ is typically chosen as the identity matrix.
4. The larger $\|\mathbf{R}_L\|$, the smaller is the gain matrix $\mathbf{G}(t)$ and the slower is the response of the system. Matrix $\mathbf{R}_L$ is especially useful for accounting for the propeller saturation in our design, and it can be tuned accordingly. $\mathbf{R}_L$ is typically chosen as the identity matrix in simulation.
5. Since the airship is inherently a low damped system, choosing effective penalties for the velocity states $\boldsymbol{\omega}$ and $\mathbf{v}$ in $\mathbf{Q}_L$ is essential for controlling possible oscillations in the time responses. The reasoning here is analogous to choosing a high derivative gain in the PD controller.

[1]The parameters used in the real flight can be finely tuned depending on the performance. But all the simulation and experimental results presented in this thesis are based on $\mathbf{F}_L = \mathbf{0}$

3.2 Control Problems for Indoor Airship Applications

Having reviewed the basic optimal control theory, we can subdivide the airship control problem into three controller design tasks. The simplest one is the hovering control, i.e., keeping the airship stationary at a given pose in the work space of the airship while rejecting perturbations including air flow disturbances, control errors caused by modelling uncertainties and linearization, as well as the airship's imbalance, which is practically unavoidable on the real system. It is pointed out that the hovering pose is not unique, and the airship must be capable of shifting from one pose to another as long as this process is quasi-stationary or quasi-time-invariant, so that the state changes between the two poses are sufficiently slow. Therefore, in designing the hovering controller, the airship can be treated as a time-invariant plant, operating at or near the equilibrium family derived in §2.3. Therefore, the infinite horizon formulation of the optimal controller design can be employed.

The second common control objective for an airship is the set-point control: with the knowledge of the current state $\mathbf{x}$, it is desired to find the optimal control input to bring the airship to the desired state $\mathbf{x}_d$. This type of maneuver is especially useful in navigation. Since there is no requirement for quasi-stationary operation, the airship can move as fast as possible and the controller must cope with the resulting nonlinearities. Fast and precise point-to-point control is the fundamental goal of the set-point controller. Meanwhile, this controller should be designed with a nonlinear methodology that can deal with the complexities omitted with the Jacobian linearization: translation control and attitude control need to be addressed together, without discarding high order dynamics in linearization. Alternatively, from the linear-system point of view, the system can be treated as a linear parameter-varying (LPV) plant as indicated in Eq. (3.4). Specifically, this plant has no exogenous signals, and hence a state-dependent state matrix can best describe the system behaviour.

Trajectory control is the last yet very important problem to tackle. This brings us to the general optimal control formulation in Eqs. (3.1–3.3). A reference trajectory $\mathbf{r}$ is prescribed along with a time constraint $t \in [t_0, T]$. The goal is to follow the reference input with

minimal tracking error and control effort. Fortunately, by using the Jacobian linearization along the reference trajectory, we are able to apply the quadratic performance index as the objective function for the tracking problem. Much of the optimal control theory discussed in §3.1 can be directly applied after solving an inverse dynamics problem and the Riccati differential equation, which will be discussed in §3.5.

In summary, the final solution to be implemented on the airship is the hybrid controller which switches among the three sub-controllers discussed earlier depending on the higher-level user specification. In the following three sections, we provide the design details and simulation results of the three controllers respectively.

3.3 Hovering Control: Gain Scheduling Approach

3.3.1 Infinite Horizon Linear Quadratic Regulator

As indicated before, the basic idea for the hovering control is to achieve perturbation rejection around the equilibrium point in a time-invariant system. Recalling the perturbation quantities defined previously in §2.3 as Eqs. (2.28–2.29), also considering the equilibrium family with $\mathbf{u}_0 = \mathbf{0}$, $\boldsymbol{\omega} = \mathbf{0}$ and $\mathbf{v} = \mathbf{0}$, we have:

$$\delta\mathbf{x}(t) \triangleq \mathbf{x}(t) - \mathbf{x}_0(t) \tag{3.18}$$

$$\delta\mathbf{u}(t) \triangleq \mathbf{u}(t) - \mathbf{u}_0(t) = \mathbf{u} \tag{3.19}$$

where $\mathbf{x}$ is the current state, $\mathbf{u}$ is the current control, $\mathbf{x}_0$ is the equilibrium state at the hovering point. From the Jacobian linearization, we already know that:

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}_0(\mathbf{x}_0)\delta\mathbf{x}(t) + \mathbf{B}_0(\mathbf{x}_0)\delta\mathbf{u}(t) \tag{3.20}$$

and the performance index can be rewritten in terms of the perturbation vectors as:

$$J = \int_{t_0}^{\infty} \Big[\delta\mathbf{x}^T(t)\mathbf{Q}_L\delta\mathbf{x}(t) + \delta\mathbf{u}^T(t)\mathbf{R}_L\delta\mathbf{u}(t)\Big]dt \tag{3.21}$$

Using the results from §3.1, it is easy to find the optimal control law:

$$\mathbf{u}(t) = -\mathbf{R}_L^{-1}\mathbf{B}_0^T\mathbf{K}\delta\mathbf{x}(t) \tag{3.22}$$

where $\mathbf{K}$ is a 12×12 matrix obtained by solving the ARE (Eq. (3.17)). Accordingly, the closed-loop system becomes

$$\delta\dot{\mathbf{x}}(t) = (\mathbf{A}_0 - \mathbf{B}_0\mathbf{R}_L^{-1}\mathbf{B}_0^T\mathbf{K})\delta\mathbf{x}(t) \tag{3.23}$$

The numerical algorithm to solve the ARE can be found for example, in [8].

The controller design presented is based on the linearized model for a specific (predefined) hovering pose so that the controller gains are computed off-line before execution on the airship. Given the computational capabilities of the modern computers, combined with the tools available in Simulink/Quarc, it is feasible to solve ARE fast enough for on-line implementation. Thus, a natural extension of the infinite horizon regulator based on the Jacobian linearization is to dynamically "schedule" the feedback gains instead of using the precomputed values. As noted earlier, since the system nonlinearity mainly arises from the orientation transformation, the scheduling variable is chosen to be the quaternion of the airship. Whether to use the current quaternion $\mathbf{q}$, or the equilibrium quaternion $\mathbf{q}_0$, however, is not an obvious choice. Using $\mathbf{q}_0$ implies the expectation that the airship does not largely deviate from the desired hovering point. Without such a constraint, using the current value of $\mathbf{q}$ may be more robust, as the controller literally works for all airship configurations. But this option requires more on-line computations since the scheduling variable needs to be updated as the attitude changes. Naturally, when the airship operates close to the equilibrium point, the two methods should produce very similar results. The last decision to make with regards to gain scheduling is how to schedule the gains. The most straightforward option is to wait until the update of the current state and then switch to the new gain value. With this method, the controller may chatter if the scheduling frequency is not sufficiently high. A common "smoothing"practice in the gain scheduling is to interpolate between the feedback gains of the two equilibrium points [54]. In our design, this would require predicting the

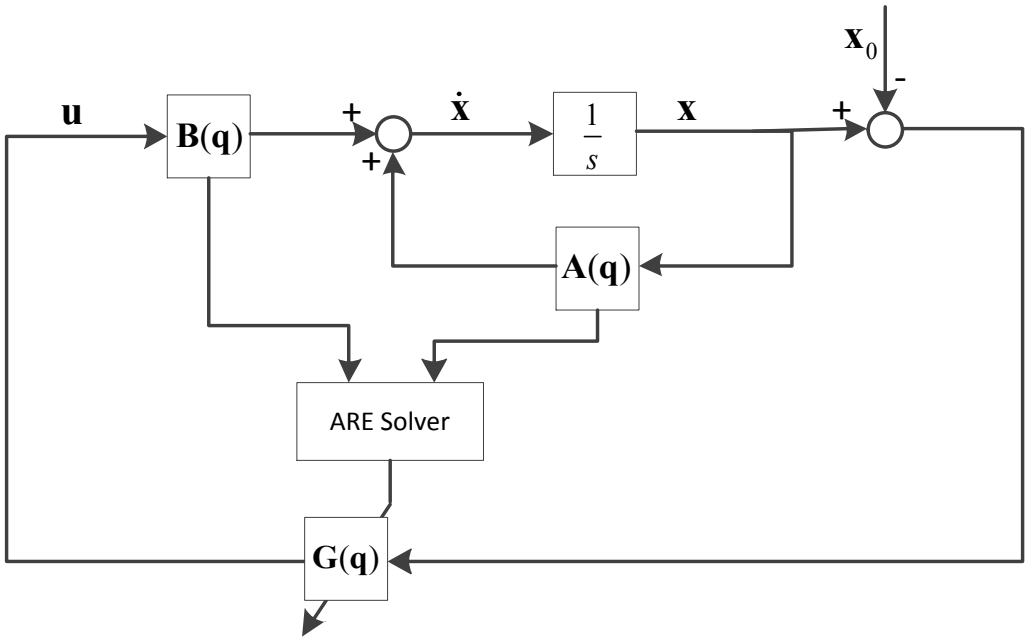


Figure 3.1 Hovering controller with gain scheduling on **q**

airship motion and then interpolating between the previous attitude and the expected attitude. Since this second option also tends to increase the amount of computation, we choose the switching approach (the scheduler is able to run stably at 10 Hz in the real-time simulation).

The final design of the hovering controller is illustrated in Fig. 3.1: a finite horizon linear quadratic regulator operating with the gain scheduling technique (referred to as LQR or hovering controller in the following text). Note, the scheduling variable **q** is produced from the current state **x**.

3.3.2 Simulation Results for Hovering Controller

Methods such as the multivariable poles-zeros analysis and the extended frequency response analysis for classical multiple-input and multiple-output (MIMO) systems can hardly be applied to our LQR controller with gain scheduling, due to the system nonlinearity and sometimes its time-variant nature. Therefore, a sequence of simulations in the time domain have been performed to discover the advantages and disadvantages of the designed LQR controller, compared with the PD controller. The controller parameters employed in the

Table 3.1 Parameters used in simulation of LQR and PD controllers

State weighting matrix $\mathbf{Q}_L$	$\mathbf{I}^{12\times12}$
Input weighting matrix $\mathbf{R}_L$	$\mathbf{I}^{12\times12}$
Gain scheduling rate	10 Hz
Controller loop frequency	100 Hz
Translational gain k_{TP} in PD control	0.8
Translational velocity gain k_{TD} in PD control	2
Rotational gain k_{RP} in PD control	0.5
Angular velocity gain k_{RD} in PD control	2
Actuator saturation boundary	$\begin{bmatrix}-0.28 & 0.48\end{bmatrix}$ N
Initial state	$\begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^T \in \mathbb{R}^{13\times1}$

simulations are listed in Table 3.1. The PD gains were carefully tuned and tested in the flight tests with the airship.

Our first focus is on the ability of the controllers to reject perturbations around a specific hovering point. To this end, white noise is added to the output of the 6 propellers with the Gaussian distribution $\mathcal{N}(0, 0.01)$. In order to clearly observe the performance of the controller with respect to the state error and the control effort, we separate the performance index in Eq. (3.21) into the integrated state index J_s and the integrated control index J_u:

$$J_s = \int_{t_0}^{\infty} \left[\delta\mathbf{x}^T(t)\mathbf{Q}_L\delta\mathbf{x}(t)\right] dt \tag{3.24}$$

$$J_u = \int_{t_0}^{\infty} \left[\delta\mathbf{u}^T(t)\mathbf{R}_L\delta\mathbf{u}(t)\right] dt \tag{3.25}$$

As shown in Fig. 3.2, J_s obtained with the PD controller is twice as high as that for the LQR result, while the control efforts are very similar, meaning that for the same input, the designed optimal controller performs with much higher precision and robustness than the PD controller.

The step input translational test, the results for which are shown in Fig. 3.3 tells the same story. In this simulation, we command a step input for the center of the airship, from the origin to $[1 \quad 0 \quad 0]^T$ at $t = 0$. The response also demonstrates how differently the LQR and PD controllers handle large reference inputs. Since this test-case involves no change of

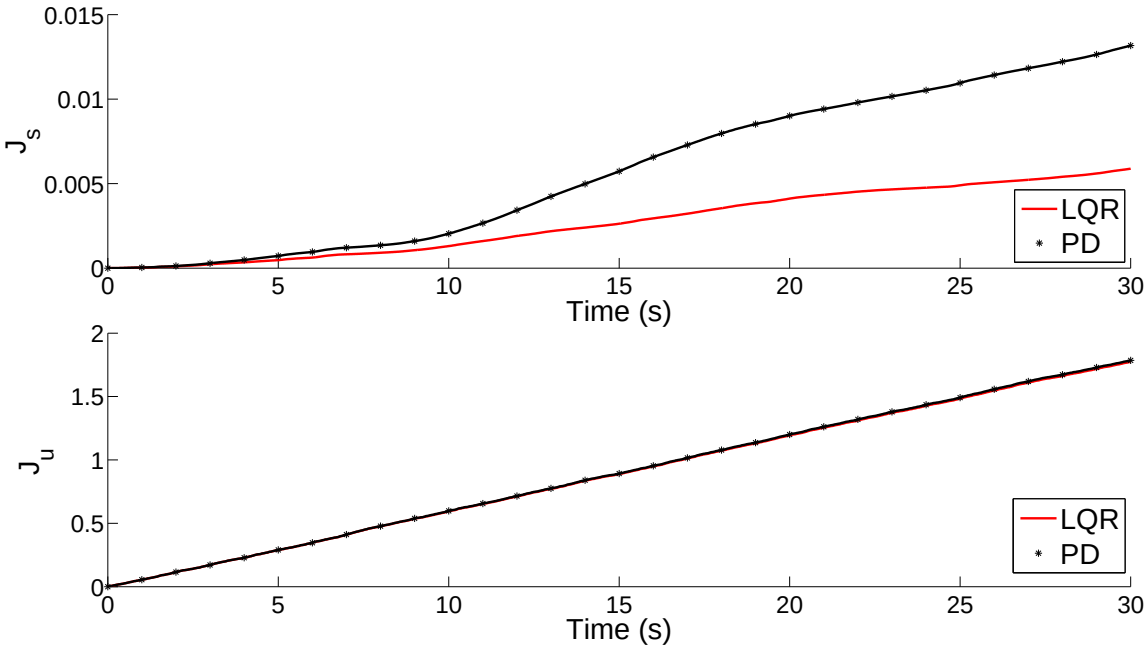
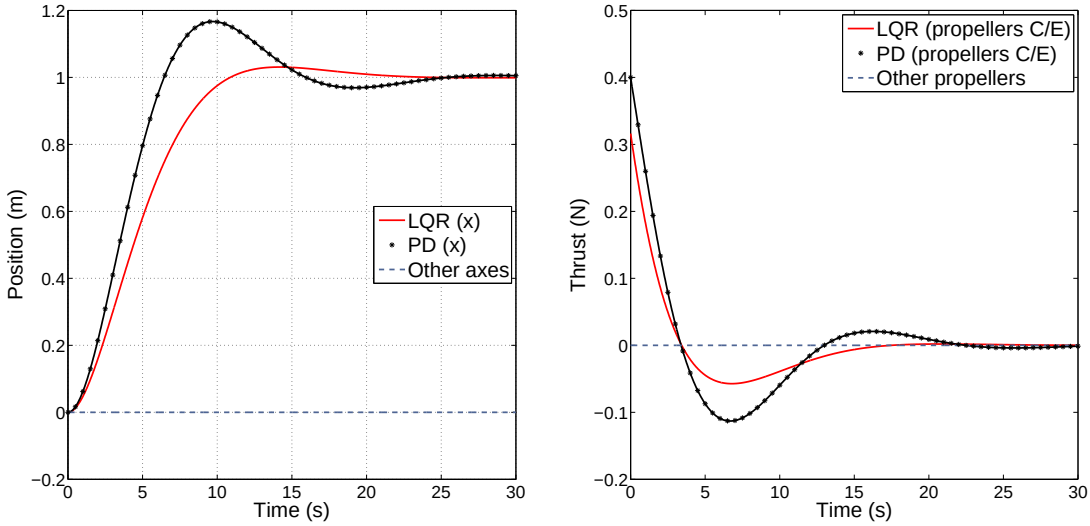


Figure 3.2 Controller state error and control effort in the presence of white noise perturbations

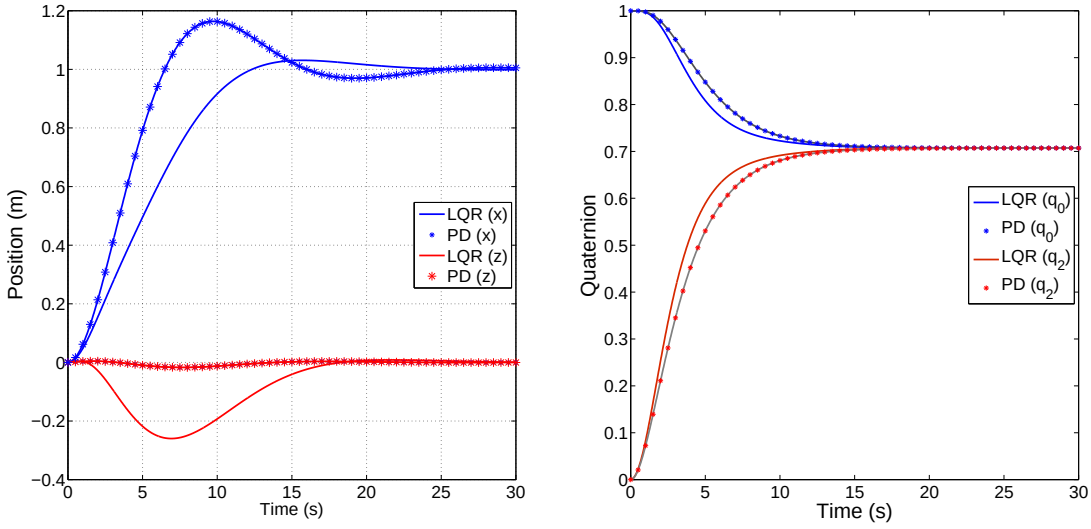
the scheduling variable **q**, the LQR controller guarantees global optimality. In time domain, this implies optimal settling time, small overshoot, along with minimal control effort, all of which are clearly observed relative to the response of the PD controller. Comprehensive simulations show that the attitude control by the LQR controller gives similar superior results, even when the quick change of the quaternion violates the quasi-stationary assumption.

Among all the maneuvers evaluated, the performance of the LQR controller suffers only for cases with multi-axes motion that combines translation and orientation control. Results for multi-axes test involving translational step change by 1 m in the X direction of the room frame XYZ and orientation step command rotating the airship by $\pi/2$ about axis Y' of the airship frame $X'Y'Z'$ are illustrated in Fig. 3.4. This test-case generates a fast variation of the scheduling variable. Since the gain scheduling approach does not predict the future state of the airship, the computed gain can not ensure globally optimal response of the airship. Moreover, as the LQR controller is based on the Jacobian linearization, when the airship operates away from the equilibrium points, Eq. (3.20) can not accurately describe the system dynamics. As a result, there will usually be a deviation on one of the axes that do not have a



(a) Step response **(b)** Control signals

Figure 3.3 Translational control using the hovering controller. Initial position: $[0 \quad 0 \quad 0]^T$, final position: $[1 \quad 0 \quad 0]^T$.



(a) Step response on the translational axes **(b)** Step response on the rotational axes

Figure 3.4 Multi-axes control using the hovering controller. Initial hovering pose: $\mathbf{q} = [1 \quad 0 \quad 0 \quad 0]^T$, $\mathbf{p} = [0 \quad 0 \quad 0]^T$, final hovering pose: $\mathbf{q} = [\cos(\pi/4) \quad 0 \quad \sin(\pi/4) \quad 0]^T$, $\mathbf{p} = [1 \quad 0 \quad 0]^T$

step input, for example, axis Z in Fig. 3.4a. Of course, this is not an issue if the airship moves slowly enough (meets the quasi-stationary assumption), but this proves that the hovering controller can not provide effective performance for set-point control tasks involving large multi-axes changes.

3.4 Set-point Control: Solving State-Dependent Riccati Equation

3.4.1 State-dependent Riccati Equation (SDRE): Overview

The State-dependent Riccati Equation (SDRE) strategy has drawn a lot of interest over the last decade, as it is an effective tool for synthesizing nonlinear feedback controllers. The SDRE strategy allows nonlinearities in the system states while also offering great design flexibility through state-dependent weighting matrices. First proposed in [50], with the theoretical foundation contributed in [21][46], this method benefited from the state-dependent coefficient (SDC) matrices, which fully capture the nonlinearities of the system. Similarly to what was done in the gain scheduling in §3.3, an ARE using the SDC matrices is solved on-line to give the suboptimal control. Since the coefficients of the ARE vary for different points in state space, the problem essentially becomes that of solving a state-dependent Riccati equation. It is shown in [21][46] that the SDRE feedback control for the infinite-horizon optimal control problem in the multivariable case is locally asymptotically stable and locally asymptotically optimal. Although relatively new to the control applications community, the SDRE strategy already has a few successful applications in the areas relevant to the application considered in this thesis, including autopilot design [45], satellite and spacecraft control [49][59], robotics [29], helicopter control and ducted fan control [61][66]. Following [22], the capabilities of the SDRE method can be summarized as below:

1. The method allows to directly specify and affect control performance through the selection of the state-dependent state and control weighting matrices $\mathbf{Q}_L(\mathbf{x})$ and $\mathbf{R}_L(\mathbf{x})$. This is inherited from its linear counterpart, the LQR control, since SDRE control can be viewed as the nonlinear extension of the infinite-horizon linear quadratic regulator.
2. The method allows to impose hard bounds on the control. This can be very helpful if one needs to consider propeller saturation in the controller design.
3. The method can satisfy state constraints, and combined state and control constraints. This is also useful, if we need to introduce state constraints to account for obstacles in the airship workspace, for example, the ceiling and the walls in a well-known indoor environment.
4. The method is able to directly handle unstable and non-minimum phase systems.
5. The method preserves the beneficial nonlinearities and utilizes the extra design degree of freedoms, available through the non-uniqueness of the SDC matrices, to enhance the performance of the system.

The drawbacks of the SDRE controller have also been considered. The major issue is that the control obtained does not guarantee global optimality with respect to the performance index, but instead is suboptimal. The choice of the SDC matrices determines whether the final control signal computed is optimal or not. However, this requires solving a partial differential equation which is as difficult as solving the HJB equation (§3.1) itself [22][19]. So far, only systems up to third order ($\mathbf{x} \in \mathbb{R}^3$) have a closed-form optimal solution [30]. Another possible concern is the computation burden of solving the ARE on-line. Fortunately, with the modern computers, this is no longer an issue for our 12-dimensional system. In case higher control rate is required, the switching technique we adopted earlier for the hovering control (§3.3) can be employed.

3.4.2 Extended Linearization

Extended linearization is also known as apparent linearization, or SDC parameterization. It is a technique for factorizing the nonlinear system into a linear structure as the controller is executed. Extended linearization produces the SDC matrices mentioned earlier, and it is the prerequisite for implementing the SDRE strategy.

Consider the deterministic, infinite-horizon nonlinear optimal regulation problem for the system below:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u}(t) \tag{3.26}$$

The system is assumed to be full-state observable, nonlinear in the state, and affine in the input. Also $\mathbf{f}(\mathbf{x}) \in C^1(\mathbb{R}^n)$ and $\mathbf{f}(\mathbf{0}) = \mathbf{0}$ and the control goal is to regulate the system to the origin such that

$$\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{0} \tag{3.27}$$

Assuming a continuous nonlinear matrix-valued function $\mathbf{A}(\mathbf{x})$ exists such that

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}(\mathbf{x})\mathbf{x} \tag{3.28}$$

where $\mathbf{A} : \mathbb{R}^{n\times n}$ is found by mathematical factorization, the system can be linearized (in extended sense) as

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}\mathbf{u}(t) \tag{3.29}$$

Here $\mathbf{A}(\mathbf{x})$ and $\mathbf{B}$ are the so-called SDC matrices, formulated for an arbitrary $\mathbf{x}$. Obviously, if $\mathbf{x}$ is a scalar, $A(\mathbf{x})$ is unique, as $A(\mathbf{x}) = f(x)/x$. For the multivariable case, for example, $\mathbf{f}(\mathbf{x}) = [x_1 \quad x_1x_2]^T$, one straightforward factorization is:

$$\mathbf{A}_1(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ 0 & x_1 \end{bmatrix} \tag{3.30}$$

and another one:

$$\mathbf{A}_2(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ x_2 & 0 \end{bmatrix} \tag{3.31}$$

A not so obvious factorization could be:

$$\mathbf{A}_3(\mathbf{x}) = \begin{bmatrix} 1 + x_2 & -x_1 \\ \frac{1}{2}x_2 & \frac{1}{2}x_1 \end{bmatrix} \tag{3.32}$$

and we notice that for $\forall \alpha, \beta \in [0, 1]$,

$$\mathbf{f}(\mathbf{x}) = \left[(1-\alpha)\,\mathbf{A}_1(\mathbf{x}) + \alpha\,(1-\beta)\,\mathbf{A}_2(\mathbf{x}) + \alpha\beta\mathbf{A}_3(\mathbf{x})\right]\mathbf{x} \tag{3.33}$$

Let $\mathbf{A}_4(\mathbf{x})$ be the coefficient matrix in Eq. (3.33). Clearly, $\mathbf{A}_4(\mathbf{x})$ is also a valid factorization. Therefore, there exists infinite number of SDC matrices obtained by simply combining the already known SDC matrices. This, as stated before, gives great flexibility in the SDRE control design process.

In extended linearization of the airship model, we have $\mathbf{x} \in \mathbb{R}^{12}$, $\mathbf{u} \in \mathbb{R}^6$ in Eq. (3.26),[2] and the dynamics of the form (3.26) can be obtained from Eq. (2.24) (in §2.2) by isolating the affine input terms, to give $\mathbf{B}$ which is the same as we obtained in Eq. 2.35. The selection of $\mathbf{A}(\mathbf{x})$ needs more consideration as far as closed-loop stability[3], and we leave it to the controller synthesis section.

3.4.3 Controller Synthesis

Given $\mathbf{A}(\mathbf{x})$ and $\mathbf{B}(\mathbf{x})$ from the extended linearization, taking $\mathbf{x} = \mathbf{0}$ as the equilibrium point, and continuing to use the infinite-horizon performance index defined in Eq. (3.5), we obtain the performance index:

$$J = \int_0^\infty \left[\mathbf{x}^T(t)\mathbf{Q}_L(\mathbf{x})\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}_L(\mathbf{x})\mathbf{u}(t)\right]dt \tag{3.34}$$

Noting that $\mathbf{Q}_L(\mathbf{x})$ and $\mathbf{R}_L(\mathbf{x})$ can vary with respect to the state $\mathbf{x}$. It is ideal to set the weighting matrices such that the feedback gain is automatically tuned down when the state deviation becomes small, and vice versa. For the simulation and for experiments, we simply choose constant $\mathbf{Q}_L$ and $\mathbf{R}_L$. Similar to the gain scheduling case, the control law is now

[2] We keep the 12 dimensional state definition in the SDRE controller design for the reason stated in §2.1.
[3] Some SDC parametrization can result in unstable $\mathbf{A}(x)$.

given via

$$\mathbf{u}(\mathbf{x}) = -\mathbf{G}(\mathbf{x})\mathbf{x} \tag{3.35}$$

The gain matrix $\mathbf{G}(\mathbf{x})$ can be obtained by mimicking the LQR formulation:

$$\mathbf{G}(\mathbf{x}) = \mathbf{R}_L^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) \tag{3.36}$$

where $\mathbf{K}(\mathbf{x})$ is now obtained by solving the SDRE:

$$\mathbf{K}(\mathbf{x})\mathbf{A}(\mathbf{x}) + \mathbf{A}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) + \mathbf{Q}_L(\mathbf{x}) - \mathbf{K}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}_L^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) = \mathbf{0} \tag{3.37}$$

The closed-loop dynamics thereby becomes

$$\dot{\mathbf{x}}(t) = \left[\mathbf{A}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{R}_L^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x})\right]\mathbf{x}(t) \tag{3.38}$$

Aside from selecting the state weighting matrix $\mathbf{Q}_L(\mathbf{x})$ and the control weighting matrix $\mathbf{R}_L(\mathbf{x})$, another major decision to make in the SDRE controller design process is to choose the SDC factorization. Although the optimal choice is still an open question being studied [19], it is prudent to choose the factorization that is most appropriate for the system and control objectives. One important consideration is the so-called state-dependent controllability matrix, given by:

$$C(\mathbf{x}) = \begin{bmatrix} \mathbf{B}(\mathbf{x}) & \mathbf{A}(\mathbf{x})\mathbf{B}(\mathbf{x}) & \dots & \mathbf{A}^{11}(\mathbf{x})\mathbf{B}(\mathbf{x}) \end{bmatrix} \tag{3.39}$$

According to the SDRE theory, the closed-loop system Eq. (3.38) has local asymptotic stability if $C(\mathbf{x})$ is full rank (12 for the airship). The straightforward factorization based on the idea used in Eq. (3.30), Eq. (3.31) and Eq. (3.33)[4] is performed by employing the additional degrees of freedom factors α, β and γ, as they function in Eq. (3.33). The SDC matrices used in the simulation are detailed in the Appendix B as Eqs. (B.1–B.11). The pointwise stability can be tested in Matlab with the controllability matrix Eq. (3.39).

Although SDRE control promises better results in handling the system non-linearity and off-equilibrium scenarios, it is still a local approach. At each time step, the SDRE gain is

[4]Since the nonlinearities in the dynamics equations are of second-order, the same idea can be used.

determined by the current state alone. Once the SDC matrices are computed, there is no ability to account for the future variations of the system. As already indicated in §3.3.2, a major drawback of the local method is that the fast regulation on the quaternion part of the state causes a large unpredicted deviation in the translational control. Fortunately, the prediction can be taken out of the optimal control loop by using a feedforward compensation, motivated to suppress the deviation as discussed next.

3.4.4 Feedforward Compensation

As shown in Fig. 3.5, for a task combining the rotational and translational movement, the vector from the initial center of the airship $\mathbf{p}_{int}$ to the desired center position $\mathbf{p}_d$ represents the ideal path for the airship. Normalizing and writing this vector in the airship frame, we have:

$$\mathbf{e} = \mathbf{R}^T(\mathbf{q})\frac{\mathbf{p}_d - \mathbf{p}_{int}}{\|\mathbf{p}_d - \mathbf{p}_{int}\|} \tag{3.40}$$

Decomposing the translational velocity $\mathbf{v}$ into the normal plane N to $\mathbf{e}$, we obtain:

$$\mathbf{v}_n = \mathbf{v} - (\mathbf{v}^T\mathbf{e})\mathbf{e} \tag{3.41}$$

As indicated in Fig. 3.5, the basic idea for the feedforward compensation is to penalize any translational velocity component in the plane N. Ideally, the larger the deviation velocity, the larger the compensating thrust should be to neutralize the deviation. Defining the control transformation matrix that connects the thrusts vector $\mathbf{u}$ with the resultant force and torque vector ($[\mathbf{f}^T \quad \mathbf{t}^T]^T$) in the airship frame,

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{t} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -L & 0 & L & 0 & 0 \\ -L & 0 & 0 & 0 & 0 & L \\ 0 & 0 & -L & 0 & L & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} = \mathbf{T}\mathbf{u} \tag{3.42}$$

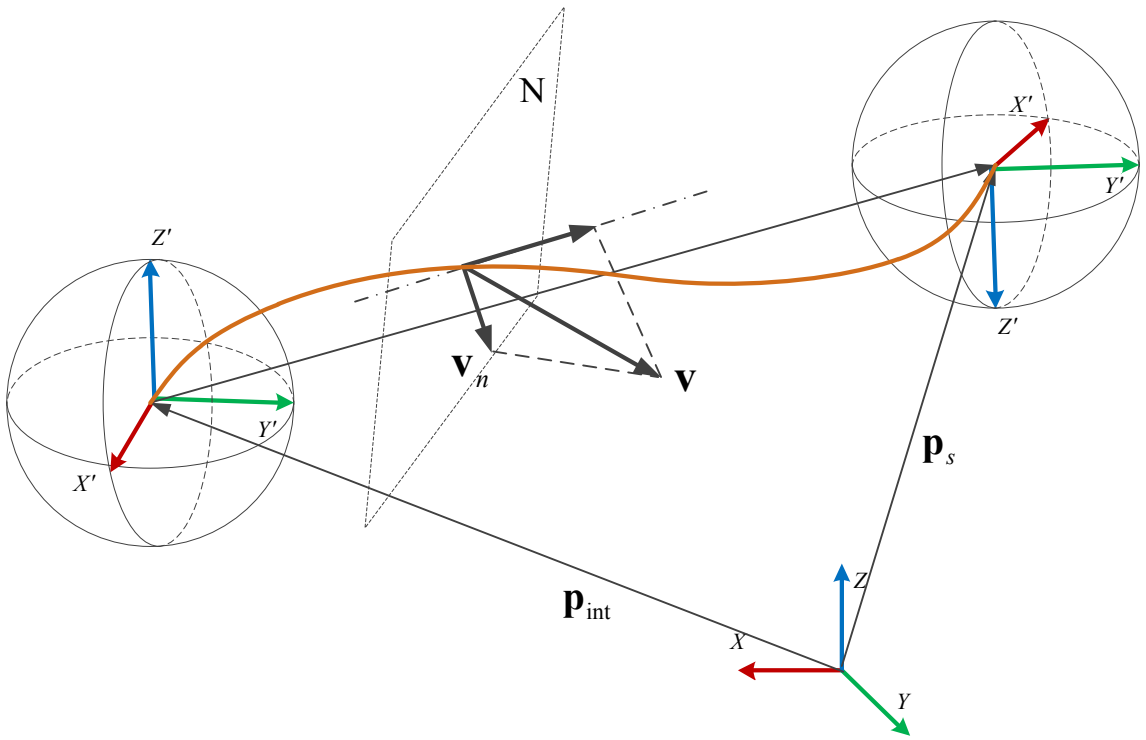


Figure 3.5 Translational velocity decomposition for the feedforward loop

where letting $L = \|\mathbf{r}\|$, i.e., the distance from the airship center to the thrust axis, and letting c denote the feedforward gain, The control law in Eq. (3.35) can be augmented with the feedforward term as

$$\mathbf{u}(\tilde{\mathbf{x}}) = -\mathbf{G}(\tilde{\mathbf{x}})\tilde{\mathbf{x}} - \mathbf{T}^{-1}\begin{bmatrix} c\mathbf{v}_n \\ \mathbf{0} \end{bmatrix} \tag{3.43}$$

In view of the above discussion, the SDRE controller with feedforward compensation has the structure shown in Fig. 3.6. First, the original formulation has to be converted to the set point frame (discussed in §3.4.5); then, the SDRE controller and the feedforward compensation act on the airship to regulate its pose. The SDRE and feedforward loops can be tuned independently.

3.4.5 Set-point transformation

One substantial difference between the SDRE design and the linear regulator used before is that in the performance index Eq. (3.34), where the state $\mathbf{x}$ is used in the SDRE instead of the

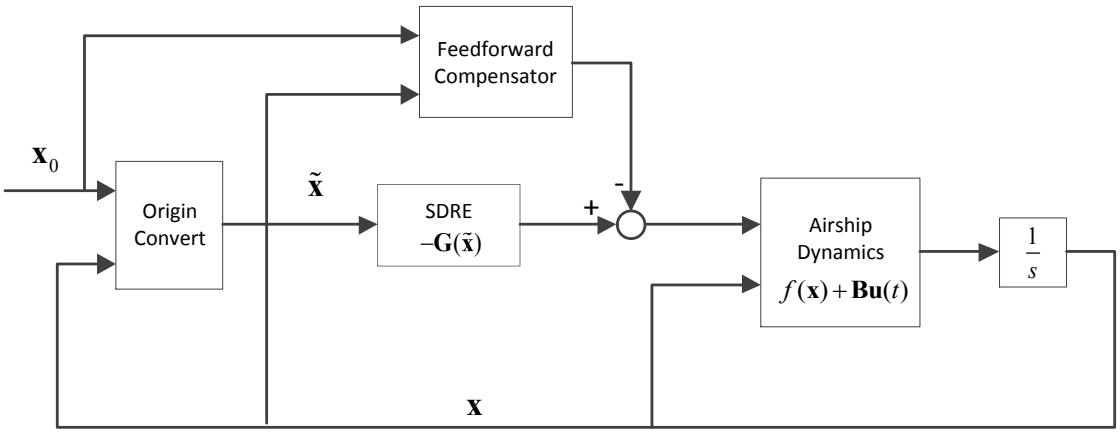


Figure 3.6 Set-point controller with forward compensation

state perturbation vector utilized in Eq. (3.21) for the LQR design. By these means, we can avoid the limitation of the Jacobian linearization, i.e., that the regulator has to operate close to an equilibrium point. However, the price paid with the SDRE method is the requirement $\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{0}$, whereas for an arbitrary set point $\mathbf{x}_d$, we must have:

$$\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}_d \tag{3.44}$$

Hence, the SDRE theory needs to be modified to suit our application.

First of all, we establish a new coordinate at $\mathbf{x}_d = \begin{bmatrix} \mathbf{q}_d^T & \mathbf{p}_d^T & \boldsymbol{\omega}_d^T & \mathbf{v}_d^T \end{bmatrix}^T$. Calculating the new state $\tilde{\mathbf{x}}$ with respect to the set point origin, we obtain:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{q}}^T & \tilde{\mathbf{p}}^T & \boldsymbol{\omega}^T & \mathbf{v}^T \end{bmatrix} \tag{3.45}$$

where

$$\tilde{\mathbf{q}} = \mathbf{q}_d' * \mathbf{q} \tag{3.46}$$

$$\tilde{\mathbf{p}} = \mathbf{p} - \mathbf{p}_d \tag{3.47}$$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} - \boldsymbol{\omega}_d \tag{3.48}$$

$$\tilde{\mathbf{v}} = \mathbf{v} - \mathbf{v}_d \tag{3.49}$$

where $\mathbf{q}_d'$ is the conjugate of $\mathbf{q}_d$. "$*$" denotes the quaternion product. (See Appendix C.)

After the conversion, we are able to use $\tilde{\mathbf{x}}$ in the SDRE controller to generate control

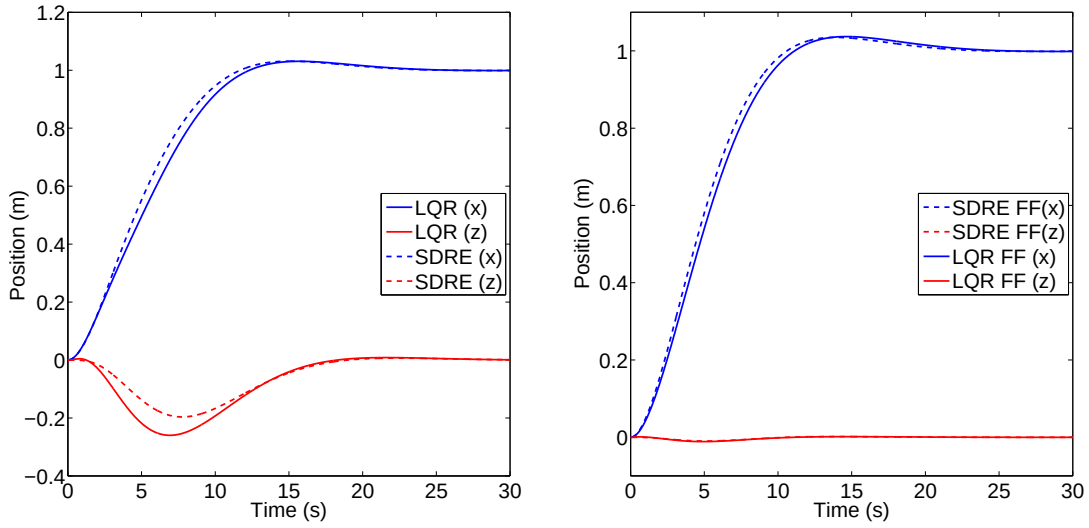
inputs of the system.

3.4.6 Simulation Results for Set-point Controller

In this section, we present the results obtained with the designed set-point controller. First, the SDRE controller is compared with the LQR controller both without feedforward compensation in a multi-axes control task. Then results of both controllers with the compensation are presented. In our implementation, determining the SDRE gain $\mathbf{G}(\mathbf{x})$ involves the same procedure as solving the ARE on-line as in Fig. 3.1. The only difference is the way to evaluate the state matrix $\mathbf{A}$, since matrices $\mathbf{B}$ obtained by the Jacobian linearization and extended linearization are the same.

Similar to §3.3.2, extensive simulations have been carried out to test the effectiveness of our design of the set-point controller. The weighting matrices are all set to identity in simulation, while the other parameters are defined as per Table 3.1 (p.42). Additional DOF factors are chosen as $\alpha = 0.5, \beta = 0.5, \gamma = 0.5$. The manuevuer is the same as we used in 3.3.2. First, we focus on the comparison between SDRE approach alone and the hovering controller presented earlier. Fig. 3.7a shows clearly that the extended linearization and the SDRE formulation of the control problem tend to alleviate, albeit not eliminate the deviation along the z axis. Meanwhile, since the SDRE approach captures the nonlinearity of the system rather than taking the first order approximation, it generates better step response. As demonstrated by the step response, the SDRE has a slightly faster settling speed. Moving onto Fig. 3.7b, the only deficiency of SDRE control, the deviation in multi-axes control is overcome by adding the feedforward compensation. By choosing a proper feedforward gain ($c = 50$ in the simulation), the controller is able to achieve even smaller deviation along the z axis than that obtained with the PD control, while maintaining all the regulation benefits of the SDRE method. As a comparison, the LQR controller with the feedforward compensation also shows nearly identical results. Therefore, it shows that the feedforward compensation is the key to solving the deviation problem for our plant. However, we still favour the

theoretical advantages of the SDRE method that the nonlinearity is handled without Jacobian linearization and equilibrium points. The simulation also proves the argument we made earlier that the optimal regulator loop and the feedforward loop can be tuned independently. Figure 3.8 shows the difference in control inputs between the SDRE alone and SDRE-feedforward schemes. The feedforward loop clearly has an impact on u_1 and u_6. It also produces a spike at $t = 10.6$s when the translational velocity v_x crosses zero and then changes direction. However, no impact has been observed on the pose response.



(a) SDRE and LQR alone

(b) SDRE and LQR with feedfowawrd compensation

Figure 3.7 Effectiveness of SDRE control with and without feedforward loop upon multi-axes step inputs. Comparison made between the SDRE controller and the LQR controller. Y axis motion is ignored since there is no significant change. Initial pose: $\mathbf{q} = [1 \quad 0 \quad 0 \quad 0]^T$, $\mathbf{p} = [0 \quad 0 \quad 0]^T$; final pose: $\mathbf{q} = [\cos(\pi/4) \quad 0 \quad \sin(\pi/4) \quad 0]^T$, $\mathbf{p} = [1 \quad 0 \quad 0]^T$

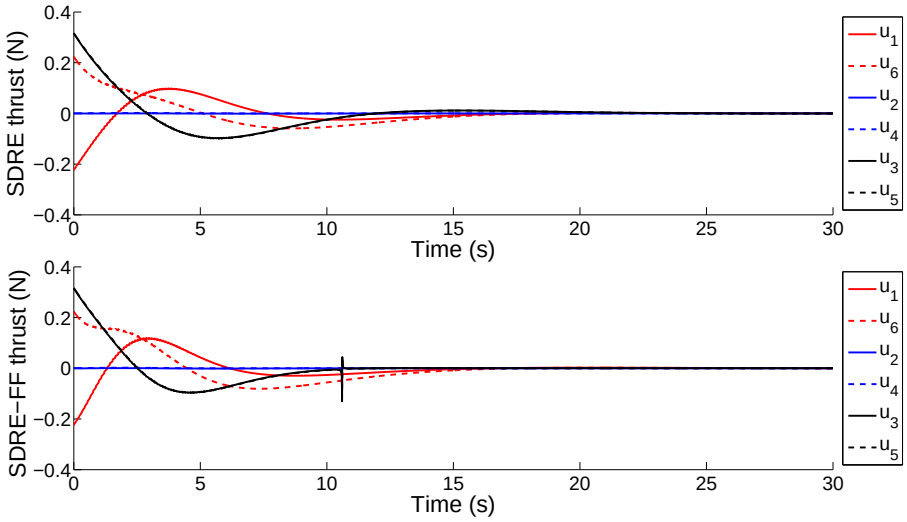


Figure 3.8 Thrust generated by the SDRE controller without and with the feedforward compensation

3.5 Reference Tracking: Continuous Linear Quadratic Problem

3.5.1 Perturbation Rejection of Continuous Reference

When it comes to the reference tracking task described in §3.1, the continuous linear quadratic (CLQR) method tends to be the best option, not only because of its capability to handle nonlinear and time-varying system, but also because of its deterministic characteristics, which allow the most demanding computation to be finished off-line before executing the feedback loop. Compared with the infinite-horizon LQR approach, CLQR also focuses on the perturbations, but in a time-varying fashion. The linearized airship model can be represented as

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}_0(t)\delta\mathbf{x}(t) + \mathbf{B}_0(t)\delta\mathbf{u}(t) \tag{3.50}$$

Letting $\mathbf{x}_0(t)$ and $\mathbf{u}_0(t)$ represent the reference state and the nominal input, the perturbations in this section are as in Eqs. (3.18–3.19) except that $\mathbf{u}_0 \neq \mathbf{0}$. Substituting perturbations on both sides of Eq. (3.5), we obtain the perturbation index:

$$J(\delta\mathbf{x}(t_0), \delta\mathbf{u}(\cdot), t_0) = \delta\mathbf{x}^T(T)\mathbf{F}_L\delta\mathbf{x}(T) + \int_{t_0}^{T} \Big[\delta\mathbf{x}^T(t)\mathbf{Q}_L\delta\mathbf{x}(t) + \delta\mathbf{u}^T(t)\mathbf{R}_L\delta\mathbf{u}(t)\Big]dt \tag{3.51}$$

The equations developed in particular Eqs. (3.13) (3.15) (3.16) in §3.1 are still valid for this perturbation index. It should be noted that the Riccati equation now becomes a matrix differential equation Eq. (3.13), which can be solved by using the Runge-Kutta integration backwards. As well, because the integration process does not require any special provisions for and overdetermined system, the 13-dimensional state definition for the airship can be retained ($\mathbf{x} \in \mathbb{R}^{13}$). The only issue left is how to determine the nominal input $\mathbf{u}_0$ based on the reference trajectory. Obviously, this is an inverse dynamics problem which can be solved by using the airship dynamics Eq. (2.24). The workflow of reference tracking can thereby be outlined as in Table 3.2.

Table 3.2 Workflow of the reference tracking of the airshp

1. Design the continuous airship trajectory $\mathbf{x}_0(t)$ within the boundary $\|\boldsymbol{\omega}\| \in [-\omega_{\min}, \omega_{\max}]$, $\|\mathbf{v}\| \in [-v_{\min}, v_{\max}]$. The maximum and minimum values are defined based on flight records.
2. Establish the nonlinear system dynamic equation. Obtain the Jacobian matrices $\mathbf{A}_0(t)$ and $\mathbf{B}_0(t)$
3. Solve the inverse dynamics problem as per Eq. (2.14), Eq. (2.15) and Eq. (2.23) to obtain $\mathbf{u}_0(t)$. If $\{\sup \mathbf{u}_0(t) > u_{\max} \vee \inf \mathbf{u}_0(t) < u_{\min}\}$ go back to step 1, and choose a smaller acceleration value.
4. Solve the differential Riccati equation backwards as per Eq. (3.13), starting from Eq. (3.15).
5. Determine the feedback gain $\mathbf{G}(t)$ as $\mathbf{G}(t) = \mathbf{R}_L^{-1}\mathbf{B}_0^T(t)\mathbf{K}(t)$
6. Initialize the real time controller with $\mathbf{K}(t)$, $\mathbf{x}_0(t)$, $\mathbf{u}_0(t)$. Start flight: controller input is $\mathbf{x}(t)$; controller output is computed by $\mathbf{u}(t) = \mathbf{u}_0(t) - \mathbf{G}(t)\delta\mathbf{x}(t)$

3.5.2 Extended Performance Index

Before presenting simulation results, in order to quantify the performance of the designed controller, we define the following performance indices (PIs) as an extension to the definition fo the objective function of the optimal control problem Eq. (3.1):

Definition 1 (Worst-case orientation error)**.** Let

$$\mathbf{R}_e = \mathbf{R}_a^T \mathbf{R}_r \tag{3.52}$$

represent the "error" rotation matrix where $\mathbf{R}_a$ is the orientation matrix of the actual attitude, and $\mathbf{R}_r$ represents the reference attitude. At every instant of time (or sampling time) t, we calculate the multi-axes orientation error as the angle of $\mathbf{R}_e$:

$$\phi(t) = \cos^{-1}\left[\frac{1}{2}\left(trace\left(\mathbf{R}_e(t)\right) - 1\right)\right] \qquad \phi \in [0, \pi] \tag{3.53}$$

The worst-case orientation error is defined by

$$w_\phi \triangleq \sup_{t\in[0,T]} \phi(t) \tag{3.54}$$

Definition 2 (Average orientation error)**.**

$$\bar{w}_\phi \triangleq \frac{1}{T}\int_0^T \phi(t) \tag{3.55}$$

Definition 3 (Worst-case translation error)**.** The three-axes translational error is:

$$p(t) = \|\mathbf{p}_a(t) - \mathbf{p}_r(t)\| \tag{3.56}$$

where $\mathbf{p}_a$ is the actual position vector, and $\mathbf{p}_r$ is the reference vector. The L^2 norm is used to compute the distance between two poses. The worst-case translational error is obtained by:

$$w_p \triangleq \sup_{t\in[0,T]} p(t) \tag{3.57}$$

Definition 4 (Average translation error)**.**

$$\bar{w}_p \triangleq \frac{1}{T}\int_0^T p(t) \tag{3.58}$$

Definition 5 (Power consumption index)**.** It can be proven that the power cost of the propelling system is proportional to the square of the control inputs $\mathbf{u}(t)$ generated by the controller. This is not obvious if one wants to account for the energy consumption of the armature of the DC motors and the cables between the driver circuit and the thrusters, on-board the airship. The proof is given in Appendix D. Therefore, the average energy consumption index W is defined as:

$$W \triangleq \frac{1}{T} \int_0^T \sum_{i=1}^{6} u_i^2(t) \tag{3.59}$$

With the PIs defined, we can now quantify the performance of the controllers over the whole trajectory tracking task.

3.5.3 Test Trajectory Design and Simulation

To demonstrate the six-axes manoeuvrability of the airship, a test trajectory is synthesised such that the airship's center follows a spiral trajectory while the airship itself rotates about the axis of the spiral line and the X' axis of the body fixed frame $X'Y'Z'$, as shown in Fig. 3.9. As a result, in the global frame, the airship motion resembles that of rolling up a spiral staircase. The trajectory parameters v_x, v_y, v_z, ω_x, ω_y, ω_z in the room frame are calculated as described below:

Letting r_0 denote the radius of the spiral and $\omega_z = \omega_0$ rad/s the angular rate about the spiral axis, the translational velocity of the airship center is then obtained as:

$$v_x = -\omega_0 r_0 \sin\theta \tag{3.60}$$

$$v_y = \omega_0 r_0 \cos\theta \tag{3.61}$$

$$v_z = \frac{\omega_0}{2\pi} \tag{3.62}$$

where $\theta = \int \omega_0 dt$ is the radial angle to the airship.

The translational speed up the spiral can be synthesised from the translational speed in the XY plane, v_{xy}, and the translational velocity along the Z axis v_z. If we choose there is no

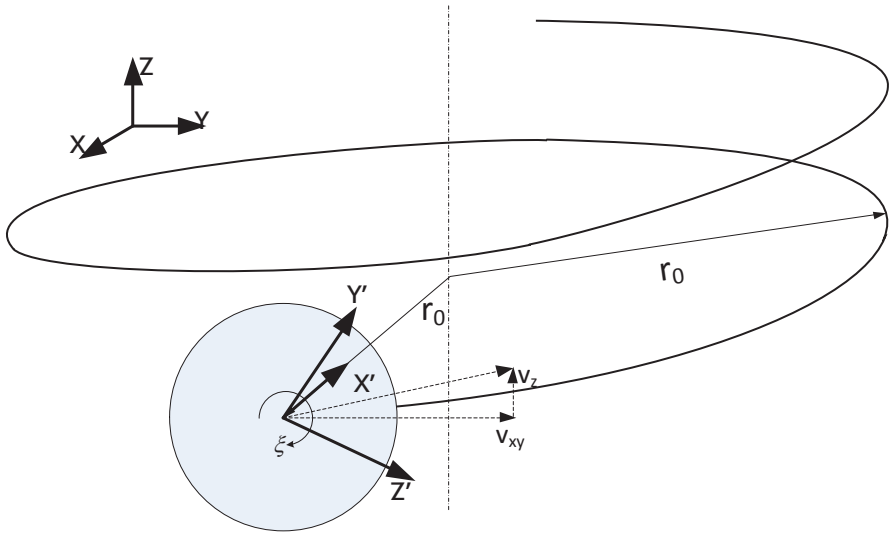


Figure 3.9 Spiral trajectory synthesis

slip on the rolling motion, the angular velocity ξ about X' axis can be derived as

$$\xi = \frac{\sqrt{(\omega_0 r_0)^2 + \left(\frac{\omega_0}{2\pi}\right)^2}}{r_a} = \omega_0 \sqrt{\left(\frac{r_0}{r_a}\right)^2 + \left(\frac{1}{2\pi r_a}\right)^2} \tag{3.63}$$

where recall r_a is the radius of the airship. Decomposing ξ as angular velocity components in the global frame, we have:

$$\omega_x = -\xi \cos\theta \tag{3.64}$$

$$\omega_y = -\xi \sin\theta \tag{3.65}$$

For the simulation results presented here, we choose $\omega_0 = 0.2$ rad/s. A 10 s constant acceleration interval is designed to make sure the airship reaches the desired speed. All the trajectory parameters v_x, v_y, v_z, ω_x, ω_y, ω_z are validated following steps 1–3 in Table 3.2 and the corresponding displacement coordinates are displayed in Fig. 3.10.

Simulations have been carried out by choosing the state weighting matrix $\mathbf{Q}_L$ as **diag** (5, 5, 5, 5, 30, 30, 30, 1, 1, 1, 1, 1, 1) $\in \mathbb{R}^{13\times13}$, and the control weighting matrix $\mathbf{R}_L$ as identity as before. Other simulation parameters are as per Table 3.1. The PD controller response/error is displayed in Fig. 3.11. Although it remains stable, the PD control shows

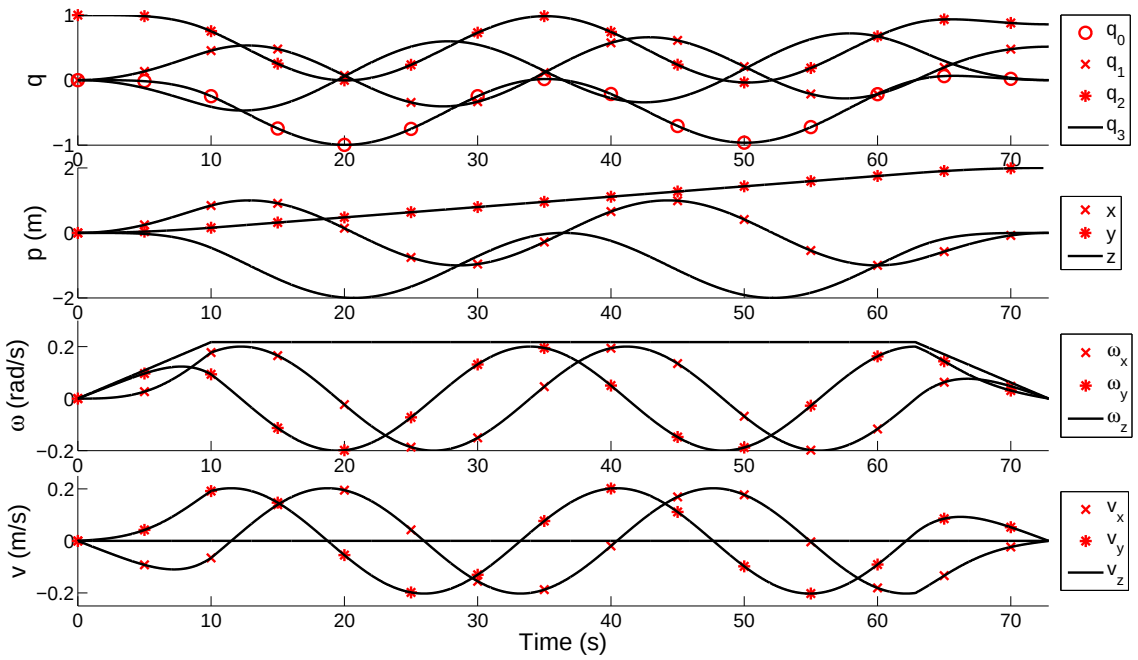


Figure 3.10 Reference trajectory in CLQR control. The quaternion, position of the airship center, angular velocity (in the body fixed frame) and translational velocity (in the body fixed frame) respectively.

very poor results, especially in the orientation control. The trajectories of angular velocities are significantly distorted, resulting in some erratic behavior and a considerable delay in the quaternion. About a five second delay also occurs in the translational response. The Z axis response is not uniform as the reference.

Focusing on the CLQR results in Fig. 3.12, it is clear that all the perturbations are under control and tend to converge to zero at the end of the maneuver. Another advantage of the CLQR control is revealed by comparing the thrust generated with both controllers. Shown in Fig. 3.13, the PD thrusts are continuous yet inefficient with respect to the changes of the desired trajectory, whereas the CLQR thrusts fully take into account those changes. For example, the discontinuities at $t = 10$s and $t = 62.84$s in Fig. 3.13b are caused by the acceleration changes in the reference. Since the CLQR controller is given this prior knowledge of the system, it is not hard to interpret why the delay is literally eliminated and the tracking error is minimized. The PD controller, on the other hand, suffering from the thruster saturation and large control effort (implying higher energy consumption), eventually fails to fulfill the tracking task.

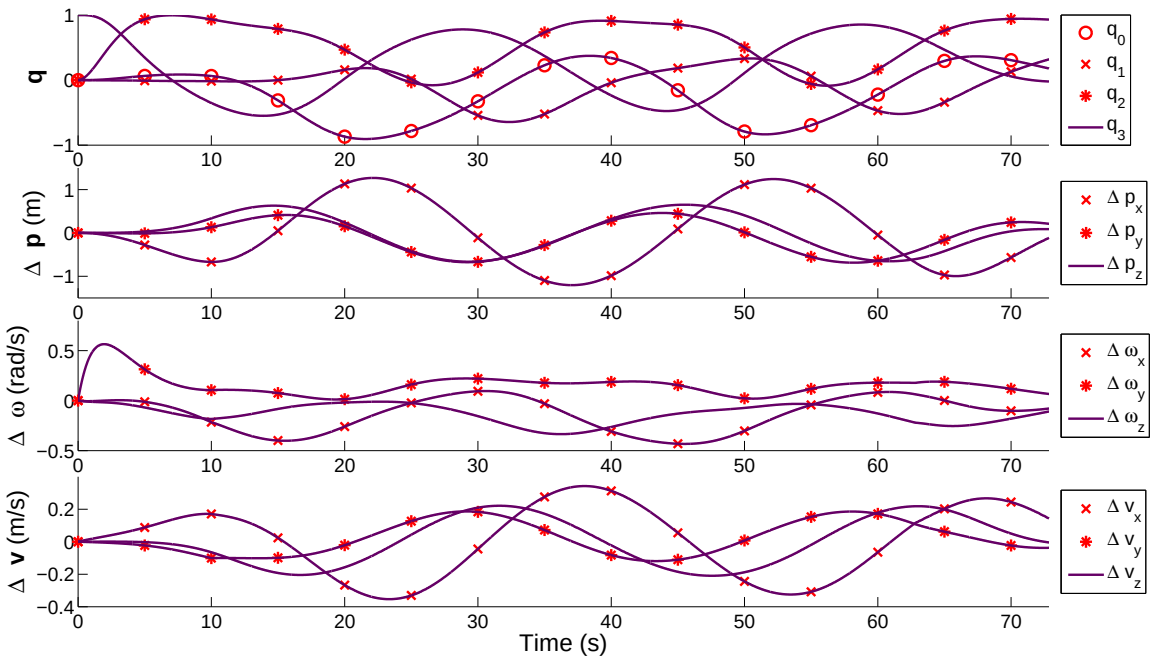


Figure 3.11 Tracking results using the PD controller

The quantitative analysis is completed by using the PIs defined in §3.5.2. The corresponding results, shown in Table 3.3 reveal the differences in the performance of the two controllers even more clearly. A parallel test is also performed after adding white noise disturbance ($\mathcal{N}(0, 0.1)$) to each thruster to emulate the uncertainty of the thrust produced in a real flight. Approximately, CLQR controller achieves an average orientation tracking error 50 times smaller and an average translational tracking error 20 times smaller than those obtained with the PD controller. On the other hand, the actuator effort of the PD controller is doubled compared to that of CLQR. Table 3.3 also indicates that in the simulation with white noise disturbances, CLQR shows robustness to the outside disturbance without compromising too much on performance.

Table 3.3 Quantitative performance comparison between CLQR and PD control

Method	$\bar{w}_p$ (m)	$\bar{w}_\phi$ (rad)	w_p (m)	w_ϕ (rad)	W ($\mathtt{N}^2 \cdot \mathtt{s}$)
PD	0.9004	1.3299	1.2777	3.1416	0.0594
CLQR	0.0430	0.0245	0.0628	0.0472	0.0313
CLQR with white noise	0.0406	0.0359	0.0657	0.0777	0.0383

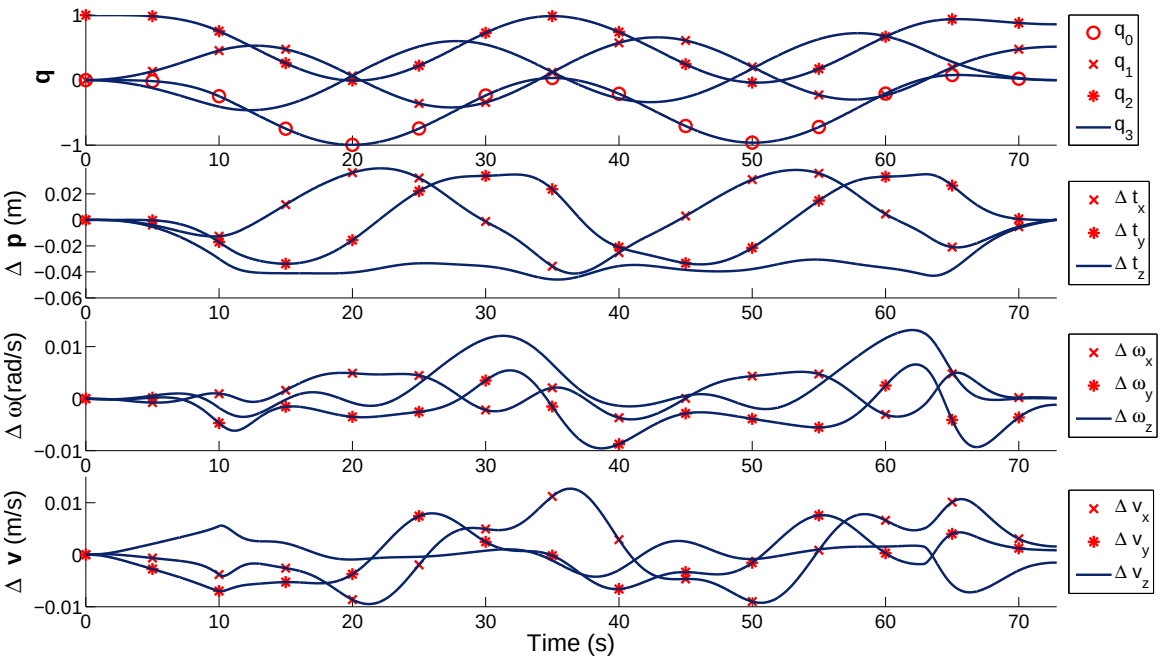
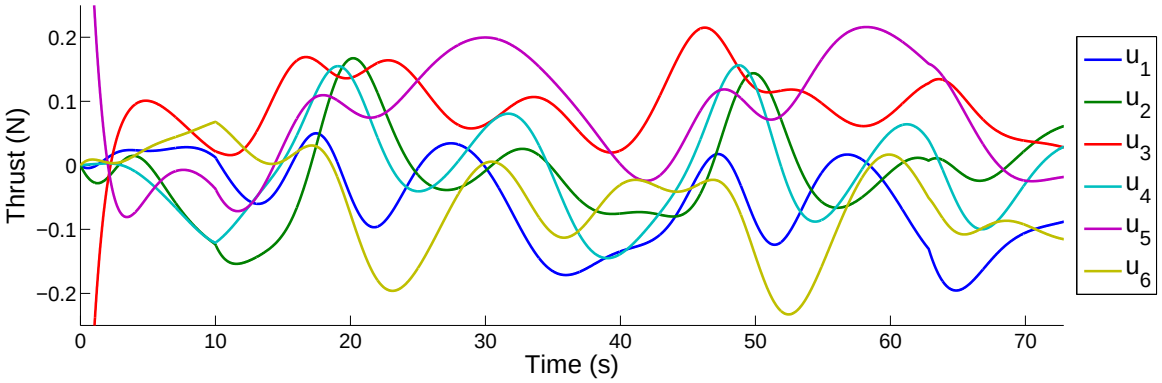


Figure 3.12 Tracking results using the CLQR controller

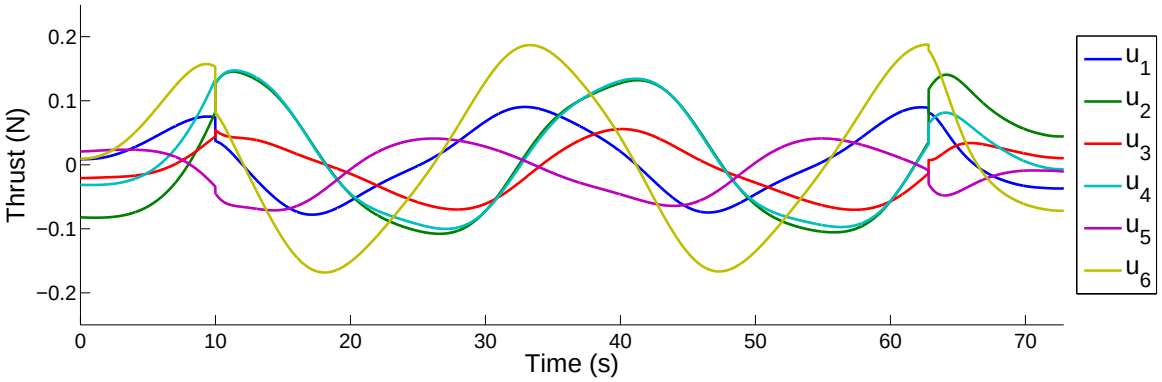
3.6 Conclusion on Optimal Control of Holonomic Airship

Three optimal controllers have been described in this chapter to accomplish different tasks of flying the holonomic airship, namely hovering control, set-point control and continuous reference tracking. All the controllers are designed so that real time implementation is feasible. The final solution on the airship would be a hybrid scheme which switches among the three controllers depending on the pilot command or the desired high-level task specification.

In the hovering control design, the airship is assumed to be a quasi-stationary plant, and we claim that a finite horizon linear quadratic regulator is the most effective design option. A gain scheduling scheme is employed to adapt to the slow change of the pose of the airship. The hovering controller exhibits great advantages over the PD controller in terms of the perturbation suppression and pure translation/orientation control, but suffers when multi-axes set-point changes in the pose are desired. Therefore, a recent approach in the nonlinear optimal control area has been employed, which is the state-dependent Riccati equation control. It is proven that the SDRE approach captures the system's nonlinearity, but



(a) Thrust by the PD controller



(b) Thrust by the CLQR controller

Figure 3.13 Comparison of thrust generated by the PD controller and the CLQR controller

it does not remedy the local optimality problem which is inherited from the optimal regulator theory. Thereby, an ad hoc feedforward loop is added to the SDRE controller in order to compensate the deviation produced on the translational axes when a rapid change of the attitude happens. It turns out that the combination of SDRE and feedforward inputs offers all the benefits of the regulator method, while bringing deviation-free results. In the end, a deterministic continuous tracker is designed for the reference tracking task. It has relatively small computation requirements, but promises superior and easily adjustable performance. On the other hand, the PD controller hardly meets the requirements of our task.

In the following chapter, we present the experimental results from the implementation of the controllers on the airship.

Chapter 4

Experiments on Airship Control

The performance of the controllers are evaluated by using extended performance indices (§3.5.2), which are defined to represent the translation and orientation errors and to quantify the energy consumption by the on-board components. Experiments have been successfully carried out based on all the controllers designed in this thesis. The purpose of the experiments is to demonstrate the improvements of the designed controllers over the PD controller. The controller parameters are set to be the same as in simulations, except for the feedforward gain c for the SDRE controller, which is tuned to 200 and the 4th to 6th diagonal terms in $\mathbf{Q}_L$, which are set to 2 due to larger deviations in real flights.

4.1 Hovering Control

Experiments have been carried out for various hovering tasks. The results of the designed LQR controller are compared with those of the PD controller in time-domain. The main criteria for a hovering task are control accuracy and the ability to reject disturbances. In a hovering task, the performance indices p, ϕ and $||\mathbf{u}||^2$ are recorded as shown in Fig. 4.1. The steady-state error of the LQR controller is smaller due to higher translational and orientation gains employed while the PD controller uses more energy but ends up with lower hovering accuracy.

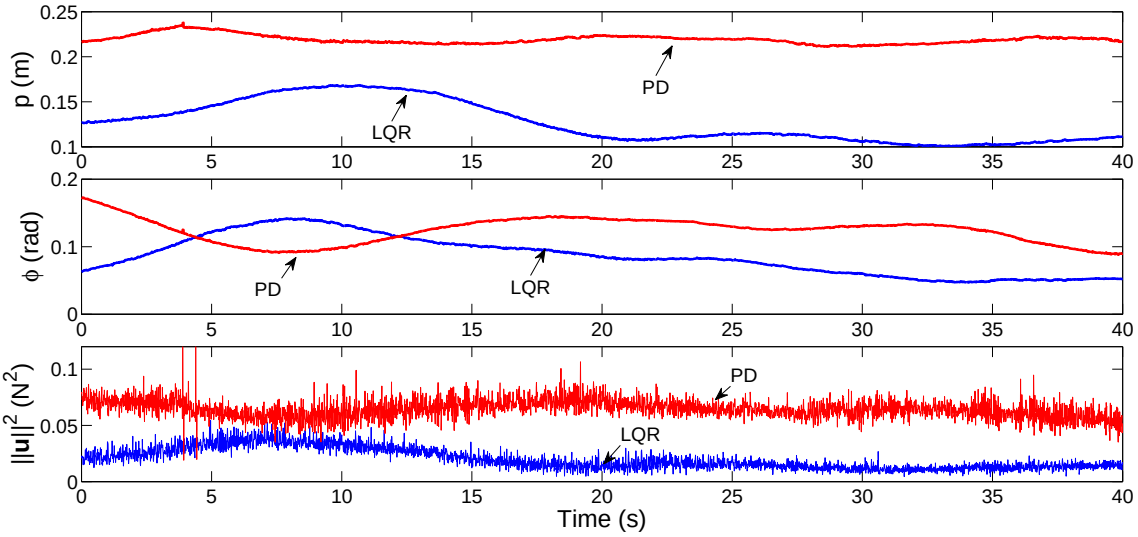
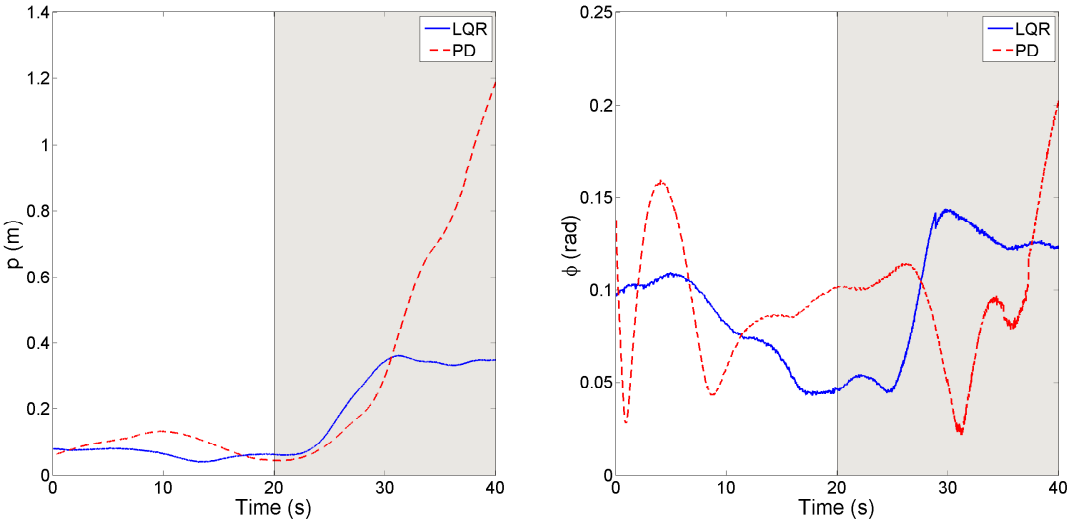


Figure 4.1 PI comparison between LQR and PD control for hovering pose: $\mathbf{q} = [1 \quad 0 \quad 0 \quad 0]^T$, $\mathbf{p} = [2 \quad 0 \quad 1.5]^T$

To emulate the worst case in a hovering task, a disturbance test is also performed, in which an industrial fan is placed 3 m away from the airship center, generating air flow over the surface of the ground, and the airship is tasked to hover 1.5 m above the ground. The translational error p and orientation error ϕ are shown in Fig. 4.2. The disturbance is enabled after $t = 20$s. It can be clearly observed that with the LQR controller, the airship is stabilized in the presence of disturbances, albeit with a larger steady-state error, while with the PD controller, the airship's attitude shows large variation and a considerable drift on translational axes occurs.

4.2 Set-point Control

Moving on to the experiments of the set-point controller, Fig. 4.4 shows results from a typical maneuver in translational control for a set-point change of 1 m in $-Y$ direction. Since this maneuver does not involve significant nonlinearity, the LQR controller has been proven to be effective in simulation as described in §3.3.2. The slight misalignment in the initial pose of the two controllers (LQR and PD) is caused by the imbalance. The results clearly demonstrate that the LQR control results in a much smaller overshoot and shorter settling



(a) Translational error **(b)** Orientation error

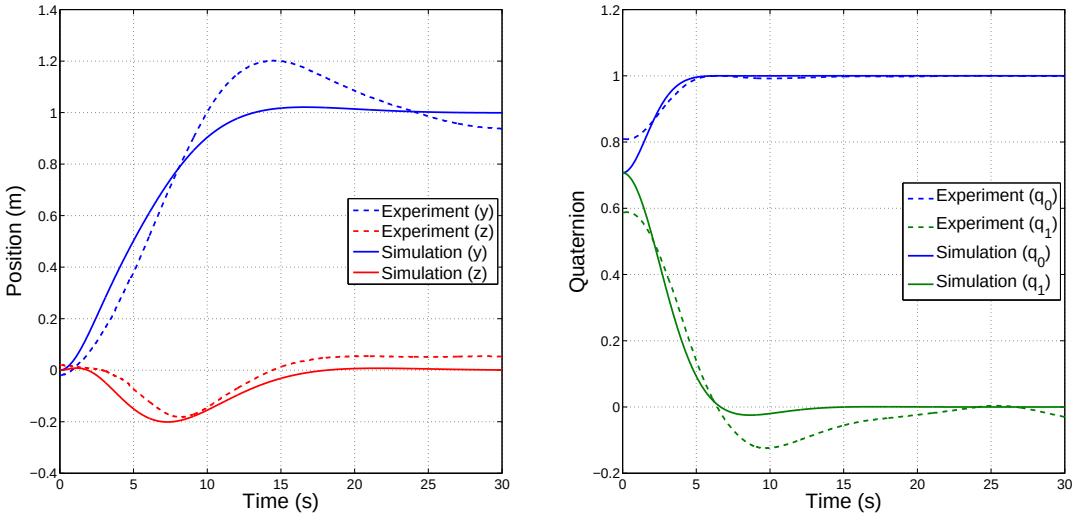
Figure 4.2 Perturbation rejection test with hovering pose: $\mathbf{q} = [1 \quad 0 \quad 0 \quad 0]^T$, $\mathbf{p} = [2 \quad 0 \quad 1.5]^T$

time in p while maintaining a lower rotational error. LQR also shows advantages over PD in terms of smaller energy consumption.

An interesting response was observed in §3.3.2 with the LQR controller for a combined maneuver with translation and rotation in the XY plane. Recall that a significant deviation in the Z-direction was observed with the LQR controller. A similar maneuver is tested experimentally, where the airship is commanded to translate 1 m in the Y direction and rotate through 90 degrees about the -X axis[1]. The results obtained with the LQR controller are shown in Fig. 4.3 for the experimental response and the simulated response. The experimental response in the Y direction is not as good as predicted in simulation with a significant overshoot observed. However, both the simulated and experiments response clearly show the significant deviation in the Z-direction for this maneuver.

General set-point control results using SDRE with the feedforward compensator are shown in Fig. 4.5. In the test, the airship is commanded to move from the initial pose defined with $\mathbf{q} = [1 \quad 0 \quad 0 \quad 0]^T$, $\mathbf{p} = [2 \quad 0 \quad 1.5]^T$ to $\mathbf{q} = [0.616 \quad 0.455 \quad 0.455 \quad 0.455]^T$,

[1]The set-point is different from the simulation because there is less ventilation disturbance along the Y axis in the lab set-up and the airship is balanced better in the final pose $\mathbf{q} = [1 \quad 0 \quad 0 \quad 0]^T$. Thus both translational and rotational responses are closer to the ideal case.



(a) Translation **(b)** Rotation

Figure 4.3 Deviation problem in multi-axes control. The axes without significant change are ignored.

$\mathbf{p} = [2 \quad -1 \quad 1.5]^T$, which represents translation of 1 m in $-Y$ direction and rotation of 104° around $[1 \quad 1 \quad 1]^T$ axis. Unlike what we have observed in simulation, significant errors in position tracking are present with both PD and SDRE controllers which we attribute to imbalance since the translation and orientation dynamics are coupled by the imbalance. Also since the airship's imbalance produces steady-state attitude error, as the thrusters try to counteract the impact of imbalance, they introduce additional disturbances. However, with the feedforward compensation, the translational error in SDRE can be remedied easily by tuning up the compensation factor c and the corresponding terms in $\mathbf{Q}_L$ that govern the translational motion. On the other hand, this results in a slower response in the attitude because of the emphasis on the translational tracking, and also amplification of the velocity noise in the control inputs which is reflected in $\|\mathbf{u}\|^2$ (See Fig. 4.5). That said, the winner of the set-point experiment is still the SDRE controller if considering overshoot, the settling time of the overall system and the steady-state error, not to mention the flexibility it offers to adjust the set-point performance by varying $\mathbf{Q}_L$ and c.

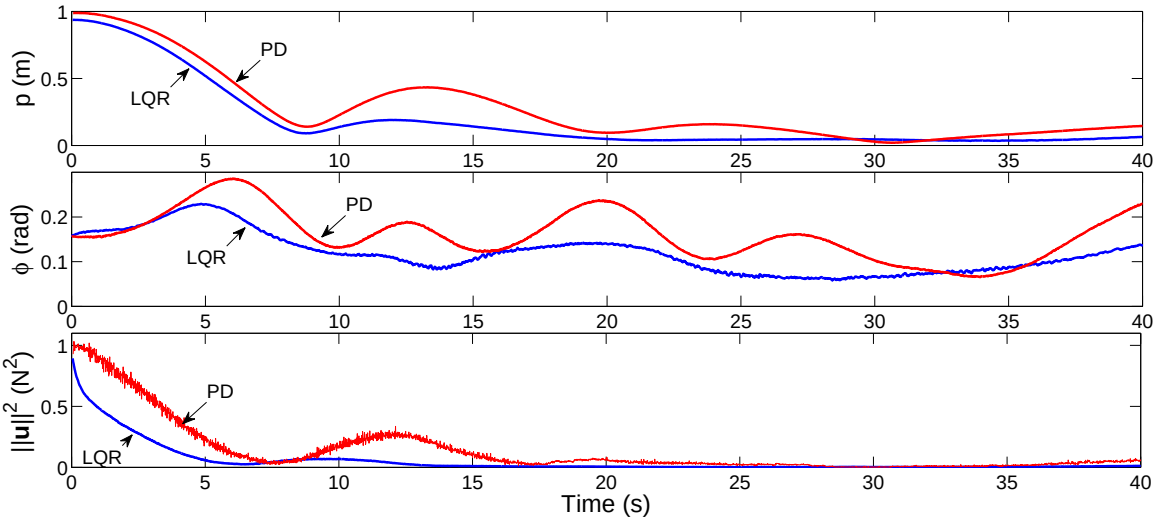


Figure 4.4 Performance comparison between LQR and PD control: translational maneuver

4.3 Trajectory Tracking

The trajectory tracking task described in §3.5.3 is also attempted. However, no satisfactory experimental result has been obtained to date due to the imbalance of the airship and slow and inaccurate response of the thrusters. Therefore, a simplified trajectory is used in the experiment: a rectangular trajectory of the airship's center is commanded in the XY plane, with no rotation. A constant acceleration process is designed to smoothly drive the airship from stationary to the maximum speed $v_m = 0.2$ m/s and then to slow it down along each edge of the rectangle. The results of continuous tracking this rectangular trajectory three times with the PD and CLQR controllers are shown in Fig. 4.6a. The CLQR result is very close to the reference except for somewhat larger errors along the right vertical edge of the rectangle where the ventilation system of the lab causes disturbances. On the other hand, the result of PD controller shows a much higher overshoot and deviation from the reference. Meanwhile, the attitude error of the PD control fluctuates during the flight as shown in Fig. 4.6b. The average error of attitude tracking is 2-3 times higher than the result of the CLQR control.

In light of the controller design goal outlined in the beginning, the optimal control problem of the holonomic airship has been solved. The next exploration, as required for all

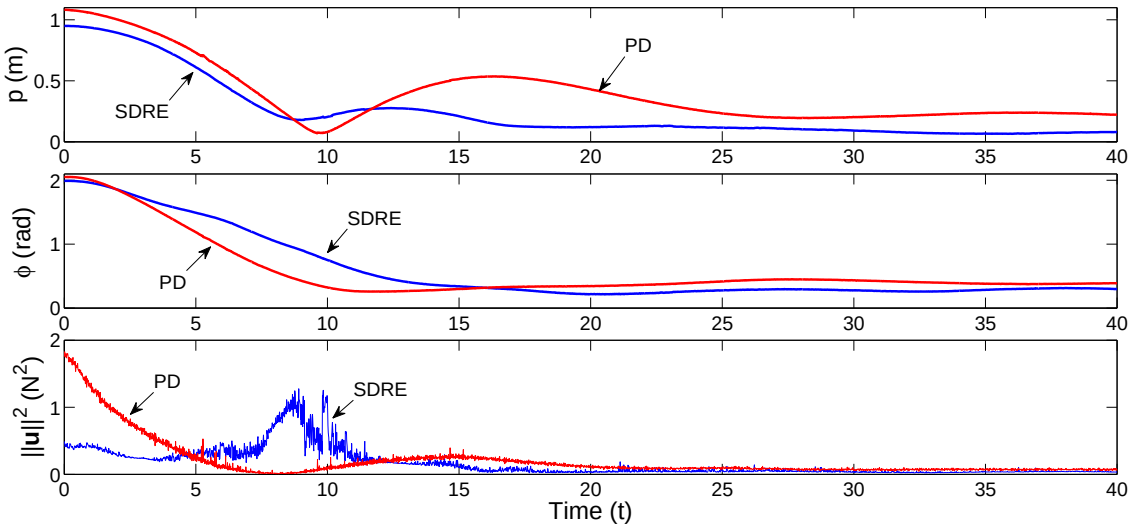
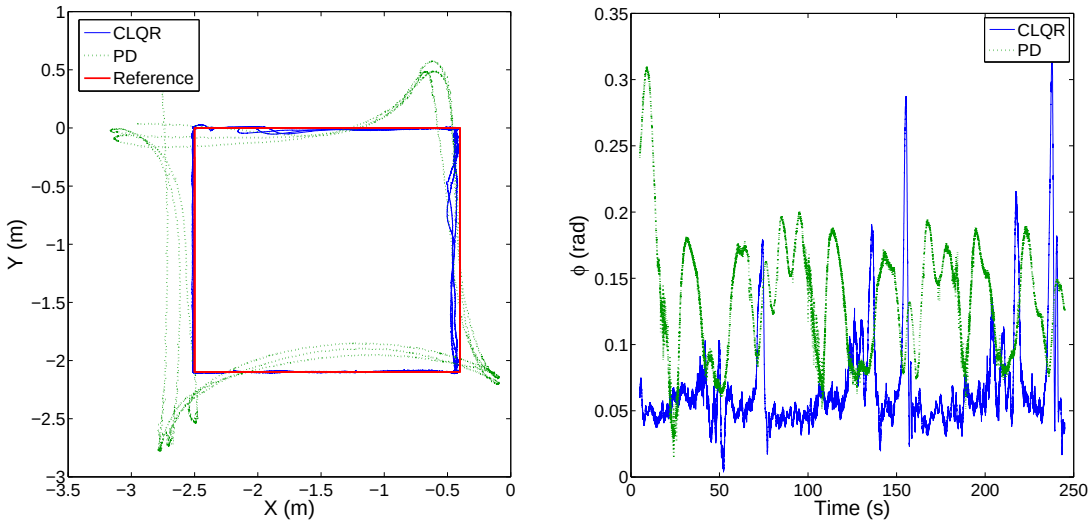


Figure 4.5 Performance comparison between SDRE and PD control: multi-axes maneuver

full-state-feedback controllers, is the estimation of the state vector $\mathbf{x}(t)$ based on all sensors available on-board or off-board.



(a) Trajectory on the XY plane

(b) Attitude error

Figure 4.6 Comparison of the trajectory tracking results of the CLQR controller and the PD controller

Chapter 5

Optimal State Estimation of Airship

A good state estimator is the key to provide reliable feedback for the control system we designed, and it also serves as the foundation for fusing all the sensors on-board and/or off-board the airship. In this chapter, a nonlinear state estimator is selected and modified for our application, and a loosely-coupled sensor fusion scheme is proposed.

5.1 Kalman Filter

Invented by Swerling and Kalman [35], the Kalman filter is probably the most well-established tool for state estimation. The Kalman filter produces estimates of the true values of state by first predicting a state with the system model. A weighted average between the predicted value and the measured value is computed by evaluating the uncertainties of both values. The most weight is given to the value with the least uncertainty. The estimates produced by the the Kalman filter are proven to be closer to the true values than any of the original measurements. Meanwhile, it shows tremendous advantages in terms of computational efficiency and the ability to combine multiple inputs with different update rates.

Kalman Filter is chosen to be the solution for the airship estimation problem for the specific reasons as follows:

1. The existing airship model is full of uncertainties. The airship imbalance, motor and sensor noise, flight disturbances and unmodelled aerodynamics all contribute to the uncertainties. A probabilistic approach in state estimation is highly needed to provide reliable feedback signal.
2. The initial state of the airship is assumed to be known in all the tasks. Although it may not be precise, it provides the initialization of the Kalman filter. Therefore, the estimation problem addressed in this work, to be more specific, is a state tracking problem rather than localization (or state "discovery" problem), which may require a global localization approach, such as Monte-Carlo algorithm [63].
3. As we have seen in §1.4.2, the low-cost IMU has a high measurement rate (100 Hz), but produces noisy measurements, while the LIDAR rate is very slow (< 5 Hz), but it contains highly reliable information. In practice, it is also quite frequent that the bluetooth communication loses data at some time steps. The Kalman filter by its nature, can combine multi-rate sensor readings and tolerate the situation of missing measurements.
4. The covariance matrix in the Kalman filter provides a tool to analyse the performance of the state estimator. The larger the covariance is, the larger the uncertainties one should expect in the filtered data. Similar to the LQR approach, the Kalman filter is easy to tune according to the actual performance.
5. The latest localization methods, such as SLAM (simultaneous localization and mapping) may have advantages in an unknown environment. However, it is still a challenge to implement those algorithms in real time, as part of the controller loop. Kalman filter is well-known for its recursive nature and low computation cost.

This chapter will be expanded in the following order: in §5.1.1, we briefly review the basics of Kalman filter along with the stochastic characteristics of linear systems; then we move to the nonlinear extension of the Kalman filter followed by a comprehensive filter design section §5.2 for our airship. The scheme of the modified Kalman filter will be

presented in §5.3 for the scenario of Vicon-IMU integration. Experimental results will be presented in the last section §5.4.

5.1.1 Stochastic Model and Kalman Filtering

There are two versions of Kalman filter: continuous version, also known as Kalman-Bucy filter and the discrete Kalman filter based on Markov chain – the recursive version. Although the Kalman-Bucy filter is recommended in [10] to work seamlessly with the controller, it requires the airship motion planned before executing the controller, and thus any deviation from the original trajectory may result in the divergence of the filter. As well, it requires solving a differential Riccati equation. From a practical point of view for our airship, the recursive version is considered a better choice.

We begin by assuming we already have a discrete and linear system of the following form[1](this is also known as the motion or process model):

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_k \tag{5.1}$$

and a measurement model:

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{5.2}$$

where $\mathbf{z}$ is the measurement vector; $\mathbf{w}$ and $\mathbf{v}$ are random signals that represent the process and measurement noise, respectively. They are assumed to be independent of each other, white and with normal probability distributions [16]:

$$p(\mathbf{w}) \sim N(0, \mathbf{Q}) \tag{5.3}$$

$$p(\mathbf{v}) \sim N(0, \mathbf{R}) \tag{5.4}$$

[1]By convention, $\mathbf{u}_k$ is used to denote the inputs of the Kalman filter, and $\mathbf{A}_k$ and $\mathbf{B}_k$ are used in the motion model. However, this notation consistency is difficult to maintain when we discretize the dynamics model in §5.2.1. Also from the system perspective, the state estimation is executed prior to the control law, and therefore, the control inputs used in the estimation always come from the previous time step. Similarly, $\mathbf{A}$ is state-dependent, thus only the matrix $\mathbf{A}$ evaluated at the previous time step is available as input to the Kalman filter algorithm. Therefore, we use $k-1$ as subscripts on the right-hand side of the model.

where $\mathbf{Q}$ is the process noise covariance and $\mathbf{R}$ is the measurement noise covariance. Both matrices describe the characteristics of the system noise and are assumed to be constant for our application. We define $\hat{\mathbf{x}}_k^-$ to be the *a priori* state estimation at step k and $\hat{\mathbf{x}}_k$ to be the *a posteriori* state estimate, given measurement $\mathbf{z}_k$ at step k. The estimate error covariance (*a posteriori*) is then defined by:

$$\mathbf{P}_k \triangleq \text{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_k) = E\left\{[\mathbf{x}_k - \hat{\mathbf{x}}_k]\,[\mathbf{x}_k - \hat{\mathbf{x}}_k]^T\right\} \tag{5.5}$$

The algorithm for the basic Kalman filter is then given in Table 5.1. Apart from the initialization, the whole algorithm is a recursive loop which can be divided into two phases: prediction phase and update phase. In the prediction phase, the system equation (model) (5.1) is used to propagate the state estimation and error covariance obtained from the last time step. In the update phase, the measurements are incorporated into the state estimation after being weighted by the Kalman gain $\mathbf{K}_k$, and the error covariance is renewed correspondingly. After each prediction/update pair, the process is repeated with the previous *a posteriori* estimates used to predict the new *a priori* estimates.

Table 5.1 Kalman filter algorithm

Initialize:		
	$\hat{\mathbf{x}}_0 = E(\mathbf{x}_0) \quad \mathbf{P}_0 = \text{cov}(\mathbf{x}_0 - \hat{\mathbf{x}}_0)$	(5.6)
Predict:		
	$\hat{\mathbf{x}}_k^- = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1}$	(5.7)
	$\mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}$	(5.8)
Update:		
	$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R})^{-1}$	(5.9)
	$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-)$	(5.10)
	$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-$	(5.11)

From the perspective of control system design, the Kalman filter and the optimal controllers designed with the linear quadratic method (§3.2) make a natural combination called

LQG (Linear Quadratic Gaussian) control [10]. As shown in Fig. 5.1, running in parallel to the real physical system, the estimated state $\hat{\mathbf{x}}$ is updated by the Kalman filter presented in Table 5.1. The measurement process $\mathbf{h}(t)$ is assumed to be linearized with a similar procedure as used to linearize the system dynamics (§2.3). Matrix $\mathbf{G}(t)$ is the feedback gain derived in accordance with the control algorithm. The controller and the basic Kalman filter algorithm are tightly coupled in the feedback loop, making it a complete solution for optimal control and estimation problems. This also represents the overall picture of the airship control system we envision in this work.

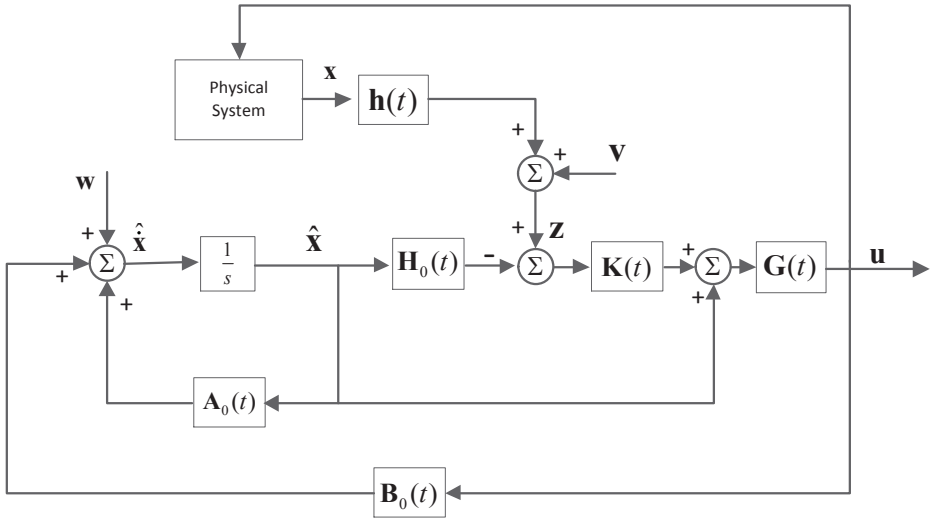


Figure 5.1 Schematic of Kalman filter combined with LQR as the Linear Quadratic Gaussian controller

5.1.2 Nonlinear Extension of Kalman Filter

In order to solve the state estimation problem for our airship, a nonlinear version of the Kalman filter is required due to the high nonlinearity in process and measurement models. Our first attempt is the so-called extended Kalman filter (EKF) [62] which utilizes the Jacobian linearization of the system model, and approximates the state distribution by transformation of Gaussian random variables. The error covariance is propagated analytically through the first-order equations. However, we encounter more and more difficulties

in implementing the Jacobian linearization when the measurement model increases complexity, specifically, for the LIDAR model. In addition, it is reported that the first-order approximation employed by the EKF may introduce large errors in the posterior mean and covariance of the transformed random variables, which may lead to sub-optimal performance and sometimes divergence of the filter [65]. Therefore, a more sophisticated approach called the unscented Kalman filter (UKF) is employed in our work [34]. The UKF addresses the aforementioned problems of the EKF by generating a minimal set of sample points (sigma points), and then propagating them through the original nonlinear system instead of a linearized one. The posterior mean and covariance are computed in the end of the algorithm based on the Gaussian assumption and are captured to the precision of the third order. Let us consider the general nonlinear system with the process and measurement models as follows:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, k) + \mathbf{w}_k \tag{5.12}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_{k-1}, k) + \mathbf{v}_k \tag{5.13}$$

We first review the so-called unscented transformation which is the foundation of the UKF. For example, if we propagate the state $\mathbf{x}$ with dimension L through the nonlinear function $\mathbf{h}$, we first form a matrix $\mathcal{X}$ of $2L + 1$ columns of sigma vectors with $\mathcal{X}_i$ denoting the ith column.

$$\begin{aligned} \mathcal{X}_0 &= \bar{\mathbf{x}} \\ \mathcal{X}_i &= \bar{\mathbf{x}} + \left(\sqrt{(L+\lambda)\mathbf{P}_x} \right)_i \qquad i = 1, 2..., L \\ \mathcal{X}_i &= \bar{\mathbf{x}} - \left(\sqrt{(L+\lambda)\mathbf{P}_x} \right)_{i-L} \qquad i = L+1, L+2..., 2L \end{aligned} \tag{5.14}$$

where $\bar{\mathbf{x}}$ is the mean of random variable $\mathbf{x}$ and $\mathbf{P}_x$ is the covariance of $\mathbf{x}$. Symbol $(\sqrt{})_i$ denotes the ith column of the matrix square root computed, for example, using Cholesky decomposition [31]. As well, $\lambda = \alpha^2(L + \kappa) - L$ is a scaling parameter; α determines the spread of the sigma points around $\bar{\mathbf{x}}$ and is usually set to a small positive value; κ is a secondary scaling parameter which is usually set to $3 - L$. Each sigma point is propagated through the state and measurement function independently, for example, the measurement

function **h**:

$$\mathcal{Z}_i = \mathbf{h}(\mathcal{X}_i) \qquad i = 0, 1..., 2L \tag{5.15}$$

and the mean and covariance for **z** are approximated using a weighted sample mean and covariance of the posterior sigma points:

$$\bar{\mathbf{z}} \approx \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Z}_i \tag{5.16}$$

$$\mathbf{P}_{\mathbf{z}} \approx \sum_{i=0}^{2L} W_i^{(c)} \left[\mathcal{Z}_i - \bar{\mathbf{z}}\right] \left[\mathcal{Z}_i - \bar{\mathbf{z}}\right]^T \tag{5.17}$$

The weights $W_i^{(m)}$ and $W_i^{(c)}$ are calculated as per the following equations:

$$\begin{aligned} W_0^{(m)} &= \lambda/(L+\lambda) \\ W_0^{(c)} &= \lambda/(L+\lambda) + (1 - \alpha^2 + \beta) \\ W_i^{(m)} &= W_i^{(c)} = 1/\left[2(L+\lambda)\right] \quad i = 1, 2..., 2L \end{aligned} \tag{5.18}$$

where β is used to incorporate prior knowledge of the distribution of **x**. For Gaussian distributions, $\beta = 2$ is optimal [65]. The complete UKF algorithm used in this thesis is given in Table 5.2. Note, the vector used to generate the sigma points is the state augmented by the process noise and the measurement noise:

$$\mathbf{x}^a = \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \\ \mathbf{v} \end{bmatrix} \tag{5.19}$$

and the compound sigma points are then defined as:

$$\mathcal{X}^a = \begin{bmatrix} (\mathcal{X}^x) \\ (\mathcal{X}^w) \\ (\mathcal{X}^v) \end{bmatrix} \tag{5.20}$$

Once the algorithm is initialized, similar to the classical Kalman filter, it will maintain a recursive loop from Eq. (5.25) to Eq. (5.35), for $k \in \{1, 2..., \infty\}$. The basic flow of the

algorithm can still be expressed in the classical form [42]:

$$\hat{\mathbf{x}} = (\text{prediction of } \mathbf{x}_k) + \mathbf{K}\left[\mathbf{z}_k - (\text{prediction of } \mathbf{z}_k)\right] \tag{5.21}$$

The major difference between the EKF and the UKF is the technique to make the prediction and to determine the Kalman gain $\mathbf{K}$. In the UKF, these are accomplished through the unscented transformation shown earlier. In addition, the algorithm we adopt in Table 5.2 has another advantage, i.e., unlike some versions of UKF [62], it does not require measurement noise to be additive (zero-mean). Therefore, the sensor bias obtained in calibration can be easily incorporated into the filter (by Eq. (5.23)) without introducing any extra dimensions in the states and modifying the motion/measurement model to incorporate the bias.

5.2 Filter Design

Following the UKF algorithm in Table 5.2, this section will focus on implementation details for our airship. First, the airship dynamics is discretized to fit the recursive form of the Kalman filter. Next, the attitude estimation problem is specified for our airship and resolved.

5.2.1 Model Discretization

Due to the recursive feature of the Kalman filter, our airship model (Eq. (2.24)) can not be directly employed in the prediction phase Eq. (5.26). A conversion is inevitable to incorporate the dynamics into the discrete Markov chain. The system requiring this conversion is sometimes referred to as a sampled-data system [57], since the dynamics are described by a continuous-time differential equation, but the input, generated by a digital computer, only changes at discrete time instants. Recalling the nonlinear process we assumed earlier in Eq. (5.12) and the system formulation established in the SDRE design in Eq. (3.29), we have:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{w}_c(t) \tag{5.36}$$

Table 5.2 Unscented Kalman filter algorithm

Initialize:

$$\hat{\mathbf{x}}_0 = E(\mathbf{x}_0) \quad \mathbf{P}_0 = \text{cov}(\mathbf{x}_0 - \hat{\mathbf{x}}_0) \tag{5.22}$$

$$\hat{\mathbf{x}}_0^a = E(\mathbf{x}_0) = [\hat{\mathbf{x}}_0^T \quad E(\mathbf{w}^T) \quad E(\mathbf{v}^T)]^T \tag{5.23}$$

$$\mathbf{P}_0^a = \text{cov}(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a) = \begin{bmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix} \tag{5.24}$$

Generate sigma points:

$$\mathcal{X}_{k-1}^a = \left[\hat{\mathbf{x}}_{k-1}^a \quad \hat{\mathbf{x}}_{k-1}^a \pm \sqrt{(L+\lambda)\mathbf{P}_{k-1}^a}\right] \tag{5.25}$$

Predict:

$$\mathcal{X}_k^{x-} = \mathbf{f}\left(\mathcal{X}_{k-1}^x, \mathcal{X}_{k-1}^w\right) \tag{5.26}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^m \mathcal{X}_{i,k}^{x-} \tag{5.27}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^c \left[\mathcal{X}_{i,k}^{x-} - \hat{\mathbf{x}}_k^-\right]\left[\mathcal{X}_{i,k}^{x-} - \hat{\mathbf{x}}_k^-\right]^T \tag{5.28}$$

Measure:

$$\mathcal{Z}_k^- = \mathbf{h}\left(\mathcal{X}_k^{x-}, \mathcal{X}_{k-1}^v\right) \tag{5.29}$$

$$\hat{\mathbf{z}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Z}_{i,k}^- \tag{5.30}$$

Update:

$$\mathbf{P}_{zkzk} = \sum_{i=0}^{2L} W_i^c \left[\mathcal{Z}_{i,k}^- - \hat{\mathbf{z}}_k^-\right]\left[\mathcal{Z}_{i,k}^- - \hat{\mathbf{z}}_k^-\right]^T \tag{5.31}$$

$$\mathbf{P}_{xkzk} = \sum_{i=0}^{2L} W_i^c \left[\mathcal{X}_{i,k}^{x-} - \hat{\mathbf{x}}_k^-\right]\left[\mathcal{Z}_{i,k}^- - \hat{\mathbf{z}}_k^-\right]^T \tag{5.32}$$

$$\mathbf{K} = \mathbf{P}_{xkzk}\mathbf{P}_{zkzk}^{-1} \tag{5.33}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \tag{5.34}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}\mathbf{P}_{zkzk}\mathbf{K}^T \tag{5.35}$$

where $\mathbf{w}_c$ is the noise of the sampled-data system with a covariance of $\mathbf{Q}_c(t)$. The solution for $\mathbf{x}(t)$ at some arbitrary time t_k is given by [57]:

$$\mathbf{x}(t_k) = e^{\mathbf{A}(t_k - t_{k-1})}\mathbf{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau)} \left[\mathbf{B}(\tau)\mathbf{u}(\tau) + \mathbf{w}_c(\tau)\right] d\tau \tag{5.37}$$

Assuming a zero-order hold (such as the one used in an A/D converter) is applied to the system, we have $u(t) = u_{k-1}$ for $t \in [t_{k-1}, t_k]$; in other words, the control signal is piecewise constant. Equation (5.37) is discretized as:

$$\mathbf{x}_k = \mathbf{\Phi}_{k-1}\mathbf{x}_{k-1} + \mathbf{\Gamma}_{k-1}\mathbf{u}_{k-1} + \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau)}\mathbf{w}_c(\tau)d\tau \tag{5.38}$$

where

$$\begin{aligned} \mathbf{\Phi}_{k-1} &= e^{\mathbf{A}\Delta t} \\ \mathbf{\Gamma}_{k-1} &= \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau)}\mathbf{B}(\tau)d\tau \\ \Delta t &= t_k - t_{k-1} \end{aligned} \tag{5.39}$$

Assuming $\mathbf{w}_c(t)$ to be additive, the state propagation can be readily obtained as:

$$\mathbf{x}_k^- = \mathbf{\Phi}_{k-1}\mathbf{x}_{k-1} + \mathbf{\Gamma}_{k-1}\mathbf{u}_{k-1} \tag{5.40}$$

The error covariance is predicted with:

$$\begin{aligned} \mathbf{P}_k^- &= E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T\right] \\ &= E\left[\left(\mathbf{\Phi}_{k-1}\mathbf{x}_{k-1} + \mathbf{\Gamma}_{k-1}\mathbf{u}_{k-1} + \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau)}\mathbf{w}_c(\tau)d\tau - \hat{\mathbf{x}}_k\right)\left(\cdots\right)^T\right] \\ &= \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1} \end{aligned} \tag{5.41}$$

where $\hat{\mathbf{x}}$ is the expectation of the state and $\mathbf{Q}_{k-1}$ is defined by

$$\mathbf{Q}_{k-1} = \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau)}\mathbf{Q}_c(\tau)e^{\mathbf{A}^T(t_k-\tau)}d\tau \tag{5.42}$$

Determining $\mathbf{\Phi}$, $\mathbf{\Gamma}$ and $\mathbf{Q}$ can be tedious and difficult for our 13 dimensional time-variant system. A series expansion approach based on Eq. (5.39) and Eq. (5.42) is proven to be efficient and accurate [23]:

$$\mathbf{\Phi} = \mathbf{I} + \mathbf{A}\Delta t + \frac{1}{2!}\mathbf{A}^2\Delta t^2 + \frac{1}{3!}\mathbf{A}^3\Delta t^3 + \cdots \tag{5.43}$$

$$\mathbf{\Gamma} = \left[\mathbf{I}\Delta t + \frac{1}{2!}\mathbf{A}\Delta t^2 + \frac{1}{3!}\mathbf{A}^2\Delta t^3 + \cdots\right]\mathbf{B} \tag{5.44}$$

Note that in the SDRE formulation of the system $\mathbf{B}$ is constant. Also, in our real-time implementation, Δt is set to 0.01 s so that the estimation is computed at the rate of the controller. The sensors' update rates are usually 1–10 times larger than Δt. Simulations of the estimation have shown that a first-order approximation of $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ is sufficient for our desired precision, since Δt is very small. Therefore, we obtain the discrete prediction model for the UKF in the simple form:

$$\mathbf{x}_k^- = (\mathbf{I} + \mathbf{A}_{k-1}\Delta t)\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1}\Delta t \tag{5.45}$$

$$= \mathbf{x}_{k-1} + \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})\Delta t \tag{5.46}$$

$$e^{\mathbf{A}(t_k-\tau)} \approx \mathbf{I} \qquad \text{for } \tau \in [t_{k-1}, t_k] \tag{5.47}$$

$$\mathbf{Q}_{k-1} \approx \mathbf{Q}_c(t_{k-1})\Delta t \tag{5.48}$$

5.2.2 Attitude Estimation

Another problem that needs to be resolved for our airship arises in the context of attitude estimation. It is well known that the traditional Kalman filter can only provide the unconstrained optimal solution of the linear stochastic estimation problem [35]. However, the quaternion representation of the attitude, as stated in Eq. (2.11), has a norm constraint. This constraint is violated when we take the weighted mean of the sigma points in Eq. (5.27) or absorb the innovation terms from the measurement in Eq. (5.34), since no mechanism has been established to enforce the constraint. This attitude estimation problem has been posed and discussed for years. For example, Psiaki [53] formulate this problem as a nonlinear programming problem and solve it using the Newton's method. Crassidis and Markley [24] convert the difference of quaternions into a constraint-free attitude parameterization, such as Rodrigues parameters or modified Rodrigues parameters (MRPs), and propagate the difference of attitude instead of the attitude itself, thus avoiding the constraint issue. Recently, Zanetti et al. prove that the optimal constraint estimate is equivalent to the result

from "brutal-force" normalization of the unconstrained estimate [67]. Formulated for the linear system but easily extended to the nonlinear case, their results greatly simplify the solution of our attitude estimation problem. Following Eq. (5.34) in the UKF, we introduce the correction via normalization:

$$\hat{\mathbf{x}}_k^{q*} = \frac{1}{||\hat{\mathbf{x}}_k^q||}\hat{\mathbf{x}}_k^q \tag{5.49}$$

where $\mathbf{x}^q$ denotes the quaternion part of the state definition (See 2.10). As well, the quaternion part of the error covariance needs to be corrected after Eq. (5.35):

$$\mathbf{P}_k^{q*} = \mathbf{P}_k^q + \frac{1}{\tilde{\epsilon}_k}\left(1 - \frac{1}{||\hat{\mathbf{x}}_k^q||}\right)^2 \hat{\mathbf{x}}_k^q(\hat{\mathbf{x}}_k^q)^T \tag{5.50}$$

where

$$\tilde{\epsilon}_k = \boldsymbol{\epsilon}_k^T \mathbf{P}_{zkzk}^{-1}\boldsymbol{\epsilon}_k \tag{5.51}$$

$$\boldsymbol{\epsilon}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k^- \tag{5.52}$$

5.3 Vicon-IMU Sensor Fusion with UKF

In this section, the process noise of the estimation model is analysed to propagate the error covariance in the predicting phase. Next, a loosely-coupled sensor fusion scheme is proposed based on Vicon-IMU integration. Characteristics of these sensors are identified in order to determine $\mathbf{R}$ to be used in the UKF. It is worth mentioning that the LIDAR-IMU integration has already been implemented and validated in simulation, but will not be presented in this thesis.

Since the Vicon system only provides the pose information of the airship, velocities were initially obtained [56] by differencing the adjacent observations and filtering the result with a low pass filter, although this still produces a noisy velocity reading with inevitable delay. Therefore, fusing the Vicon data with the IMU measurements and smoothing the inputs to the control system becomes the first task for the UKF. The designed scheme and signal routing are shown in Fig. 5.2. First of all, all parts of the state are available directly from the

Vicon system ($\mathbf{x}^p$, $\mathbf{x}^q$ and $\mathbf{x}^v$ from differencing) and IMU measurement ($\mathbf{x}^\omega$), implying the measurement matrix $\mathbf{H}$ is an identity matrix in Eq. (5.2). The position ($\mathbf{x}^p$) and orientation ($\mathbf{x}^q$) measurements are directly taken from the Vicon station at 100 Hz. As mentioned before, the acceleration data from the IMU is too noisy to be integrated, and therefore the translational velocity ($\mathbf{x}^v$) is obtained by differencing the position data from Vicon, as was done previously in [56]. On the other hand, the gyroscope data is processed by a build-in filter on the IMU and then converted to the airship frame to represent the angular velocity ($\mathbf{x}^\omega$) of the airship. The update interval of the IMU is usually larger than the value of Δt at which the UKF is running due to the transmission limit of reliable bluetooth communication. In case a faulty reading is detected, for example the Vicon measurement $\mathbf{z}_k^V$ in Fig. 5.2, the predicted state $\mathbf{x}^-$ and covariance $\mathbf{P}^-$ are passed on to the next time step without the update step of the filter.

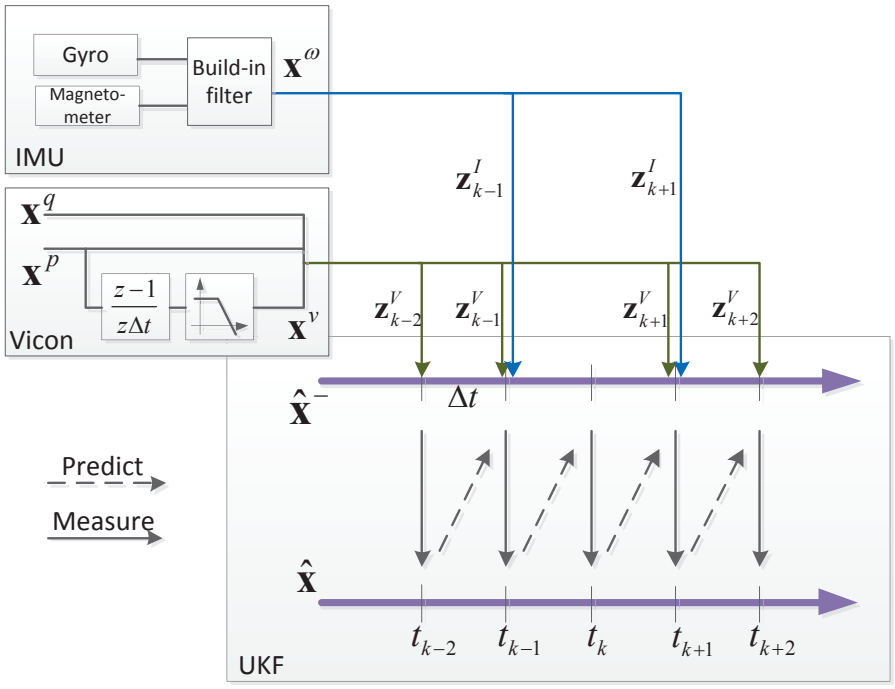


Figure 5.2 Multi-rate Vicon-IMU integration with the UKF

The next parameters to determine are the process noise covariance $\mathbf{Q}_c$ and the measurement noise covariance $\mathbf{R}$ for the Vicon-IMU integration. First focusing on the process noise

$\mathbf{w}_c(t)$ in Eq. (5.36), it can be modelled as:

$$\mathbf{w}_c(t) = \mathbf{B}(t)\boldsymbol{\beta}(t) + \boldsymbol{\eta}(t) \tag{5.53}$$

where $\boldsymbol{\beta} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\beta(t))$ is the thruster noise, and $\boldsymbol{\eta} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\eta(t))$ is the modelling uncertainty which can be identified by comparing the model-predicted value of the airship state and the actual state in test flights. Using the property of linear transformation of Gaussian random vectors [17], the covariance of thruster noise is combined with the covariance of modelling uncertainty as:

$$\mathbf{Q}_c(t_k) = \mathbf{B}(t_k)\boldsymbol{\Sigma}_\beta(t)\mathbf{B}^T(t_k) + \boldsymbol{\Sigma}_\eta(t) \tag{5.54}$$

We have already shown that $\mathbf{B}$ is independent of the state vector and time in the SDRE design (See §3.4). Therefore, $\mathbf{Q}_c$ is time-invariant if we assume that the thruster noise and modelling uncertainty are intrinsic characteristics that do not change with time. Lacking the accurate understanding of the motor and modelling errors from experiments, we take $\boldsymbol{\Sigma}_\beta$ and $\boldsymbol{\Sigma}_\eta$ as diagonal matrices scaled by variances σ_β^2 and σ_η^2 such that:

$$\boldsymbol{\Sigma}_\beta = \sigma_\beta^2 \mathbf{I}^{6\times 6} \qquad\qquad \boldsymbol{\Sigma}_\eta = \sigma_\eta^2 \mathbf{I}^{13\times 13} \tag{5.55}$$

On the other hand, the measurement noise is relatively easy to obtain from the experimental tests with the airship. Figure 5.3 demonstrates the multi-axes measurement noise recorded in a typical flight. During the flight, the airship is commanded to do various hovering and set-point maneuvers, and the zero-mean noise is extracted from the ground truth trajectory which is obtained by smoothing the Vicon data. It can be clearly observed that the Vicon system provides pose measurements with a standard deviation on the order of 10^{-3} (m for the position), while the angular velocity measurements from the IMU suffer from the noise generated by the PWM control and can only achieve a standard deviation of approximately 0.03 rad/s. Averaging data from several experiments like this, the measurement covariances can be obtained as detailed in Appendix E.

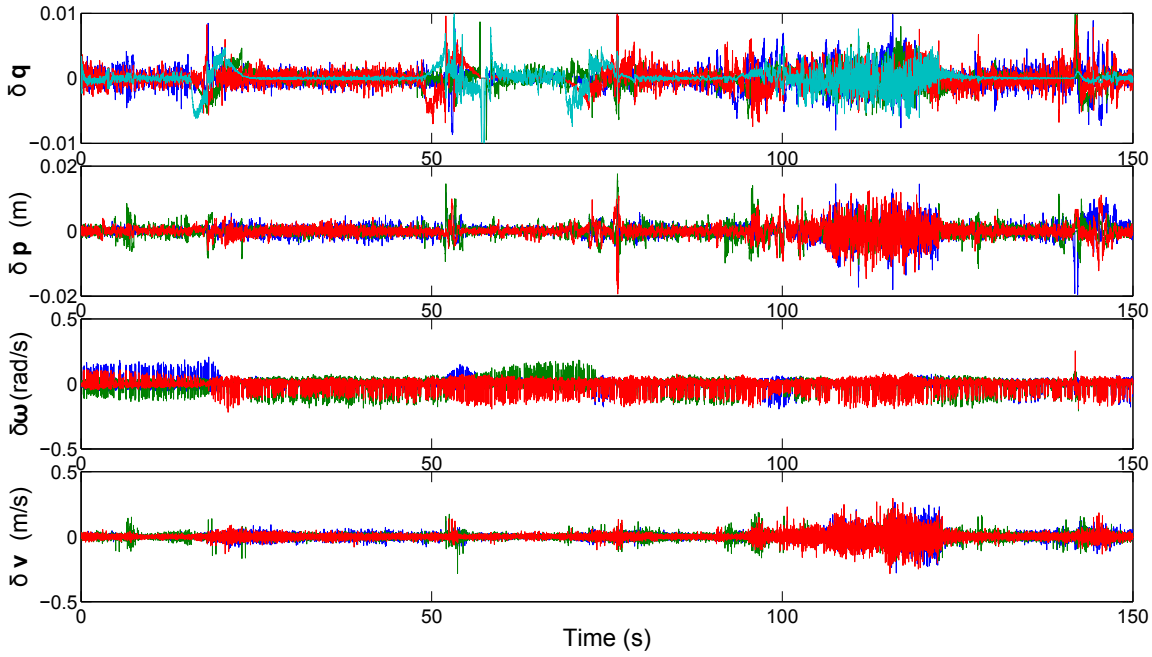


Figure 5.3 Measurement noise of Vicon-IMU integration

5.4 Results of State Estimation and LQG control

In this section, experimental results will be first presented for the Vicon-IMU integration, and then, the designed CLQR controller will be integrated with the UKF to form the aforementioned LQG controller (§5.1.1). In the experiments, the parameters in the process model are selected by estimating the errors from the recorded flight data as $\sigma_\beta = 0.1$ N and $\sigma_\eta = 0.1$ (m for position, m/s for translational velocity and rad/s for angular velocity). In real flights, we are able to update the Vicon system and the IMU at the highest sampling rate (100 Hz). The other parameters are kept as default in the UKF design (§5.2): $\alpha = 1e^{-3}$, $\beta = 2$.

In the Vicon-IMU integration, the UKF essentially functions as a low-pass filter that incorporates the IMU measurements based on the dynamics model we developed in §2.2. Focusing on the angular velocity results, a sample flight cropped from a hovering task is shown in Fig. 5.4, where we compare the raw angular velocities with UKF processed velocities. Clearly, the velocity observation results are much smoother and a quantitative analysis indicates the covariance to be approximately 100 times smaller than that of the

raw measurements. On the other hand, since the Vicon system already has a very small

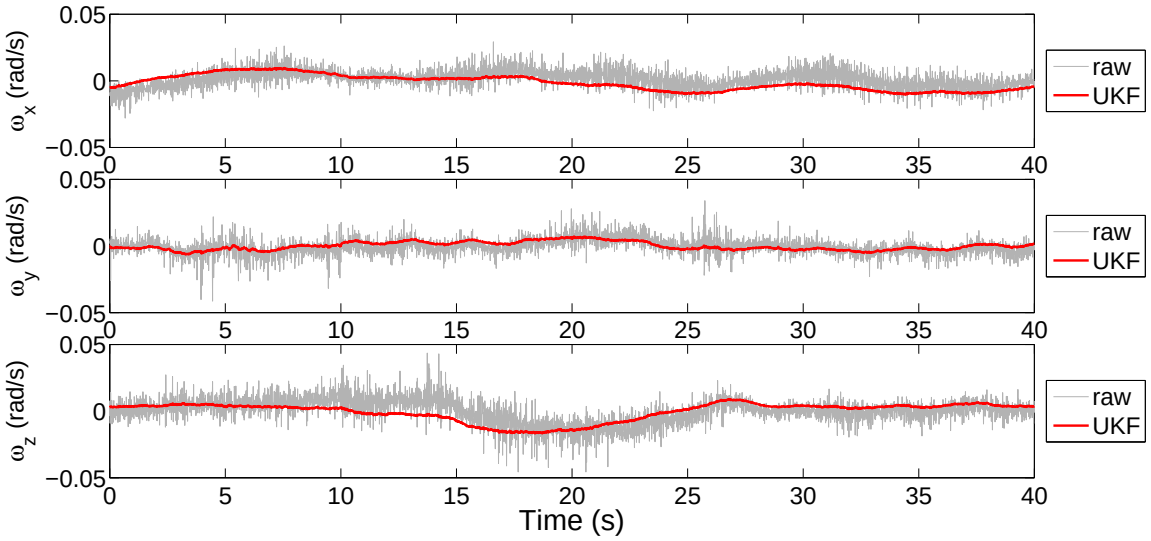


Figure 5.4 Comparison of angular velocities, filtered and raw data from the IMU

measurement uncertainty, the pose does not show substantial differences compared with the input (Vicon) data.

With the successful implementation of the state estimator, the linear quadratic Gaussian control is the natural combination of the estimator and the controllers designed in §3.2. For a preliminary test, the parameters and trajectory employed in §4.3 are used to control the airship to follow a rectangle using CLQR control. The position error, attitude error and the control inputs (§3.5.2) from the second cycle along the rectangular path are recorded as shown in Fig. 5.5. No substantial improvements on the tracking precision are observed after adding the UKF to the feedback loop. However, the energy consumption is dramatically reduced since the velocity feedback signals are much smoother than those without the Kalman filtering. Therefore, the combination of the designed controller and estimator is recommended in the future flight.

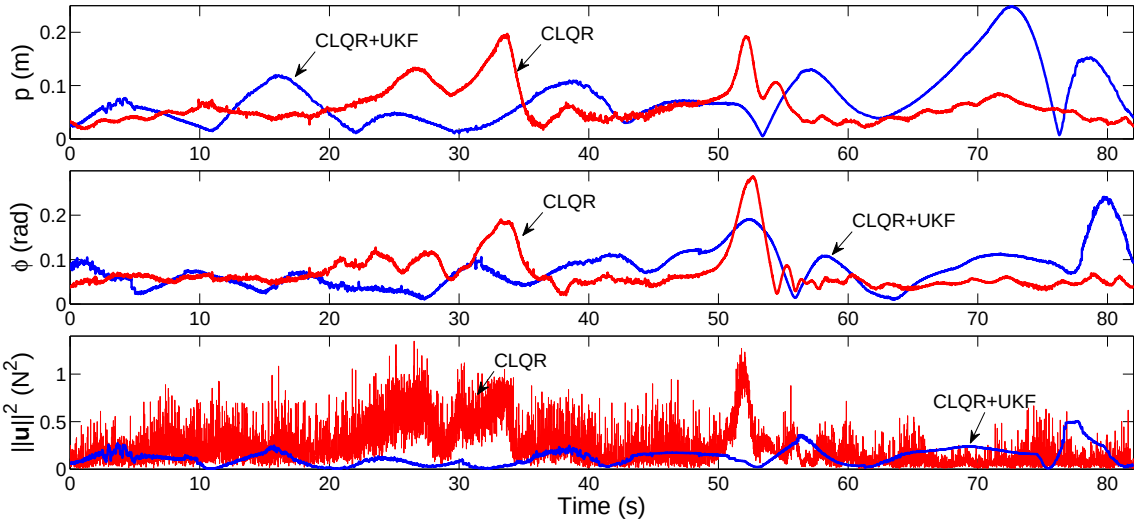


Figure 5.5 Comparison of CLQR control with and without the UKF

Chapter 6

Conclusion

6.1 Conclusion on Airship Control

Three full-state feedback controllers have been proposed to fulfill the requirements for flying an indoor holonomic airship in real-time, namely, hovering control, set-point control and continuous reference tracking. In the hovering control design, the airship is assumed to be a quasi-stationary plant. An infinite horizon linear quadratic regulator (LQR) operating in a gain scheduling manner is employed. The proposed controller doubles the pose control accuracy of the PD controller in the presence of white noise disturbances, while exhibiting faster step response. Also the airship exhibits more robust behavior than with the PD controller in the experiments. However, the LQR controller is limited by the Jacobian linearization which ignores the nonlinear dynamics and requires the plant to operate closely to the equilibrium points. As well, only local optimality can be guaranteed with LQR for multi-axes set-point control. A controller based on the state-dependent Riccati equation (SDRE) and ad hoc feedforward compensation is synthesized to tackle this issue. Both simulation and experimental results show that this method fully captures the system nonlinearity and combines the advantages of the quadratic regulator and the PD controller in terms of deviation suppression. Given the airship motion is slow, we found that the LQR controller with the

feedforward compensation can produce a good performance for our indoor application. Lastly, a continuous tracker (CLQR) is dedicated to rejecting all disturbances along an arbitrary reference trajectory. With little computation cost, the proposed controller achieves an average orientation tracking error 50 times smaller and an average translational tracking error 20 times smaller than those obtained with the PD controller with the designed trajectory in simulation. In experiment, although the same maneuver could not be reproduced due to the limitation of the thrusters, a simplified trajectory test clearly demonstrates the advantages of the CLQR controller.

6.2 Conclusion on State Estimation

An optimal state estimator designed with the unscented Kalman filter is implemented in this work. The purpose is to track the airship state for the feedback loop and other navigation tasks by fusing information from the on-board (an inertia measurement unit and a laser range finder) and/or off-board (an infra-red based motion capture system) sensors with different update rates. Therefore, a multi-rate sensor fusion scheme has been employed. The UKF algorithm is also modified to work with the dynamic model of the system and to support attitude estimation using quaternion. Experiments validate the algorithm proposed for airship state estimation and show great improvement (covariance 100 times smaller after filtering) of the angular velocity reading. Experiments of combing the optimal controller and the state estimator also show considerable improvements on energy efficiency.

6.3 Suggestions for Future Work

6.3.1 Improvements to Hardware

In regards to the status quo of our airship, as addressed before, perfect balancing and neutrally buoyancy are not achievable in practice. In the controller validation experiments,

the airship's balance has to be adjusted every 5–10 minutes due to the helium leakage, as otherwise the controller has to expand most of the actuator authority to fight the steady-state errors and it is difficult to make a consistent comparison between the different controllers. On the other hand, the dynamic response of our current propellers is not sufficiently fast. The motor calibration shown in Fig. 1.3 is based on steady state outputs of the propellers and the thrust readings in the calibration are very noisy even in a steady state. In fact, the true thrusts generated by the propellers in a real flight remains unknown. To sum up, stronger and faster propellers are highly recommended for the next upgrade of the airship. Ideally, they should come with build-in velocity feedback so that the calibration could be done at the rotor speed-thrust level and the actuating loop is closed by the speed controller. Thus, we could measure the thrusts applied to the airship from the rotor speed instead of predicting them from the control command.

To that end, the RC transmitter and the on-board controller also need an upgrade to support the new thrusters. Besides, we have encountered many difficulties during the experiments: the driving circuit tends to malfunction when large thrusts are commanded on multiple propellers. This may be due to the voltage fluctuation generated by the PWM control. Thus, a stand-alone power supply may be necessary for the on-board processor as well.

Meanwhile, the bluetooth and RS-232 communications have become another bottleneck in the system. Even at the maximum transmission rate, the current communication interface is still unable to carry the IMU and LIDAR signals at full speed, and package losses occur at times. In addition, there is no RS-232 port left on the control station for future expansion of the system, not to mention the delay caused in the two-way communication and its impact on the controller. A desirable upgrade would be, moving the controller and estimator on-board, considering their reasonable computation requirements. As another fundamental step toward our ultimate goal – the airship as an autonomous indoor UAV, this change may not be feasible in the near future due to the payload limitations of our current system. However, it

is advisable that at minimum all communications are implemented under 802.11 standard, and hence all sensors including the Vicon system are unified under one protocol (such as TCP/IP) with a much wider bandwidth.

6.3.2 Improvements to Software

In terms of airship control, there is still room for further improvement. First of all, an integral term can be added to the optimal controller by using the augmented state proposed in [4], so that the steady-state error caused by imbalance can be effectively eliminated. However, a better long-term solution of the imbalance issue is to use the adaptive control to estimate the varying gravity center and the inertia tensor, and hence the imbalance can be accounted for by the controller. However, this can not be considered seriously until the upgrade of the actuating system and a successful acquisition of the actual thrust applied to the airship are implemented.

A promising improvement on the state estimation is to integrate a light-weight camera. Our preliminary attempt with optical flow algorithms with a wireless camera did not yield good results [63]. However, with the successful combination of the IMU and LIDAR, it is worth investigating the benefits of combining the LIDAR and a camera. Our current LIDAR-IMU integration is limited because the LIDAR has to work with landmarks and the operating range of the airship is confined. As our goal is to achieve a fully autonomous UAV capable of operating in an unknown environment, the visual information is especially useful.

References

[1] Abrego, A., Chang, I., and Bulaga, R., 2002, "Performance Study and CFD Predictions of a Ducted Fan System," AHS Aerodynamics, Acoustics, and Test and Evaluation Technical Specialist Meeting.

[2] Achtelik, M., Bachrach, A., He, R., Prentice, S., and Roy, N., 2008, "Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments," in *Robotics: Science and Systems Conference*.

[3] Allgöwer, F. and Zheng, A., 2000, *Nonlinear model predictive control*, Birkhäuser.

[4] Anderson, B. and Moore, J., 1971, *Linear optimal control*, Englewood Cliffs, N.J., Prentice-Hall.

[5] Anderson, B., Moore, J., and Naidu, D., 1990, *Optimal control: linear quadratic methods*, vol. 1, Prentice Hall Englewood Cliffs, NJ.

[6] Andrievsky, B. and Fradkov, A., 2002, "Combined adaptive autopilot for an UAV flight control," in *Control Applications, 2002. Proceedings of the 2002 International Conference on*, IEEE, **1**, pp. 290–291.

[7] Angeles, J., 2007, *Fundamentals of robotic mechanical systems: theory, methods, and algorithms*, Springer Verlag.

[8] Arnold III, W., 1983, "On the numerical solution of algebraic Riccati equations," Ph.D. thesis, Univ. of Southern California.

[9] Astrom, K. and Wittenmark, B., 1994, *Adaptive control*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[10] Athans, M., 1971, "The role and use of the stochastic linear-quadratic-Gaussian problem in control system design," *Automatic Control, IEEE Transactions on*, **16**(6), pp. 529–552.

[11] Athans, M., 2002, "The role and use of the stochastic linear-quadratic-Gaussian problem in control system design," *Automatic Control, IEEE Transactions on*, **16**(6), pp. 529–552.

[12] Bachrach, A., He, R., and Roy, N., 2009, "Autonomous flight in unknown indoor environments," *International Journal of Micro Air Vehicles*, **1**(4), pp. 217–228.

[13] Badia, S., Pyk, P., and Verschure, P., 2005, "A biologically based flight control system for a blimp-based UAV," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, IEEE, pp. 3053–3059.

[14] Bijker, J. and Steyn, W., 2008, "Kalman filter configurations for a low-cost loosely integrated inertial navigation system on an airship," *Control Engineering Practice*, **16**(12), pp. 1509–1518.

[15] Bijnens, B., Chu, Q., Voorsluijs, G., and Mulder, J., 2005, "Adaptive feedback linearization flight control for a helicopter UAV," in *2005 AIAA Guidance, Navigation, and Control Conference and Exhibit; San Francisco, CA*, AIAA, pp. 1–10.

[16] Bishop, G. and Welch, G., 2001, "An introduction to the Kalman filter," *SIGGRAPH Course Note*.

[17] Bryson, A., 2002, *Applied linear optimal control: examples and algorithms*, Cambridge University Press.

[18] Chou, J., 2002, "Quaternion kinematic and dynamic differential equations," *Robotics and Automation, IEEE Transactions on*, **8**(1), pp. 53–64.

[19] Çimen, T., 2008, "State-dependent Riccati equation (SDRE) control: a survey," in *Proc. of the 17th IFAC World Congress, Seoul, Korea*, pp. 3761–3775.

[20] Clift, R., Grace, J., and Weber, M., 1978, *Bubbles, drops, and particles*, vol. 380, Academic press New York.

[21] Cloutier, J., D'Souza, C., and Mracek, C., 1996, "Nonlinear regulation and nonlinear H-infinity control via the state-dependent Riccati equation technique. I- Theory," in *International Conference on Nonlinear Problems in Aviation and Aerospace, Daytona Beach, FL*, pp. 117–130.

[22] Cloutier, J. and Stansbery, D., 2002, "The capabilities and art of state-dependent Riccati equation-based design," in *American Control Conference, 2002. Proceedings of the 2002*, IEEE, **1**, pp. 86–91.

[23] Crassidis, J. and Junkins, J., 2004, *Optimal estimation of dynamic systems*, vol. 2, Chapman & Hall.

[24] Crassidis, J. and Markley, F., 2003, "Unscented filtering for spacecraft attitude estimation," *Journal of Guidance Control and Dynamics*, **26**(4), pp. 536–542.

[25] Crowe, C., Sommerfeld, M., and Tsuji, Y., 1998, *Multiphase flows with droplets and particles*, CRC.

[26] Davies, J., 2009, "The aerodynamics of golf balls," *Journal of Applied Physics*, **20**(9), pp. 821–828.

[27] Eberly, D., 1999, "Quaternion algebra and calculus," Tech. rep., Geometric Tools, LLC.

[28] Erdem, E., 2001, "Analysis and real-time implementation of state-dependent Riccati equation controlled systems," Ph.D. thesis, University of Illinois at Urbana-Champaign.

[29] Erdem, E. and Alleyne, A., 2001, "Experimental real-time SDRE control of an underactuated robot," in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, IEEE, **3**, pp. 2986–2991.

[30] Erdem, E. and Alleyne, A., 2004, "Design of a class of nonlinear controllers via state dependent Riccati equations," *Control Systems Technology, IEEE Transactions on*, **12**(1), pp. 133–137.

[31] Golub, G. and Van Loan, C., 1996, *Matrix computations*, vol. 3, The Johns Hopkins University Press.

[32] Gomes, S. and Ramos Jr, J., "Airship dynamic modeling for autonomous operation," in *Robotics and Automation, IEEE International Conference on*, **4**, pp. 3462–3467.

[33] Hoag, D., 1963, "Apollo guidance and navigation: considerations of Apollo IMU Gimbal lock," Tech. rep., MIT Instrumentation Laboratory Document E-1344.

[34] Julier, S. and Uhlmann, J., 1997, "A new extension of the Kalman filter to nonlinear systems," in *Proc. AeroSense: 11th Int. Symp. Aerospace/Defense Sensing, Simulation and Controls*, pp. 182–193.

[35] Kalman, R., 1960, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, **82**(1), pp. 35–45.

[36] Kang, S., Nam, M., Kim, B., Tsubouchi, T., and Yuta, S., 2003, "A novel design and control of robotic wheeled blimp for tele-guidance," in *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, IEEE, **1**, pp. 67–72.

[37] Kawamura, H., Kadota, H., Yamamoto, M., Takaya, T., and Ohuchi, A., 2005, "Motion design for indoor blimp robot with PID controller," *Journal of Robotics and Mechatronics*, **17**(5), pp. 500–508.

[38] Kendoul, F., Fantoni, I., and Nonami, K., 2009, "Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles," *Robotics and Autonomous Systems*, **57**(6-7), pp. 591–602.

[39] Kim, T., Stol, K., and Kecman, V., 2007, "Control of 3 DOF Quadrotor Model," *Robot Motion and Control 2007*, pp. 29–38.

[40] Ko, J., Klein, D., Fox, D., and Haehnel, D., 2007, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *Robotics and Automation, 2007 IEEE International Conference on*, IEEE, pp. 742–747.

[41] Leith, D. and Leithead, W., 2000, "Survey of gain-scheduling analysis and design," *International Journal of Control*, **73**(11), pp. 1001–1025.

[42] Lewis, F., 1986, *Optimal estimation*, John Wiley & Sons.

[43] Li, Y., 2008, "Dynamics modeling and simulation of flexible airships," Ph.D. thesis, McGill University.

[44] Mayhew, D., 1999, "Multi-rate sensor fusion for GPS navigation using Kalman filtering," Master's thesis, Virginia Polytechnic Institute and State University.

[45] Menon, P., Lam, T., Crawford, L., and Cheng, V., 2002, "Real-time computational methods for SDRE nonlinear control of missiles," in *American Control Conference, 2002. Proceedings of the 2002*, IEEE, **1**, pp. 232–237.

[46] Mracek, C. and Cloutier, J., 1998, "Control designs for the nonlinear benchmark problem via the state dependent Riccati equation method," *International Journal of Robust and Nonlinear Control*, **8**(4–5), pp. 401–433.

[47] Munson, B., Young, D., and Okiishi, T., 1998, *Fundamentals of fluid mechanics*, John Wiley & Sons.

[48] Oh, S., Kang, S., Lee, K., Ahn, S., and Kim, E., 2006, "Flying display: autonomous blimp with real-time visual tracking and image projection," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, pp. 131–136.

[49] Parrish, D. and Ridgely, D., 1997, "Attitude control of a satellite using the SDRE method," in *American Control Conference, 1997. Proceedings of the 1997*, IEEE, **2**, pp. 942–946.

[50] Pearson, J., 1962, "Approximation methods in optimal control," *International Journal of Electronics*, **13**(5), pp. 453–469.

[51] Persson, M., 2007, "Aerodynamic forces on a sphere under general motion: literature review and developments," Tech. rep.

[52] Persson, M., 2007, "Closed-loop control over a neutrally-buoyant airship," Tech. rep.

[53] Psiaki, M., 2000, "Attitude-determination filtering via extended quaternion estimation," *Journal of Guidance, Control, and Dynamics*, **23**(2), pp. 206–214.

[54] Rugh, W. and Shamma, J., 2000, "A survey of research on gain-scheduling," *Automatica*, **36**(10), pp. 1401–1425.

[55] Sawatzki, O. and Zierep, J., 1970, "Das Stromfeld im Spalt zwischen zwei konzentrischen Kugelflächen, von denen die innere rotiert," *Acta Mechanica*, **9**(1), pp. 13–35.

[56] Sharf, I., Laumonier, B., Persson, M., and Robert, J., "Control of a fully-actuated airship for satellite emulation," in *Robotics and Automation Video Proceedings, 2008. IEEE International Conference on*, IEEE, pp. 2218–2219.

[57] Simon, D., 2006, *Optimal state estimation: Kalman, H∞ and nonlinear approaches*, LibreDigital.

[58] Singh, S., Pachter, M., Chandler, P., Banda, S., Rasmussen, S., and Schumacher, C., 2000, "Input-output invertibility and sliding mode control for close formation flying of multiple UAVs," *International Journal of Robust and Nonlinear Control*, **10**(10), pp. 779–797.

[59] Stansbery, D. and Cloutier, J., 2000, "Position and attitude control of a spacecraft using the state-dependent Riccati equation technique," in *American Control Conference, 2000. Proceedings of the 2000*, IEEE, **3**, pp. 1867–1871.

[60] Suzukia, T., 2006, "A memory-based PID controller for indoor airship robot," *Intelligent autonomous systems 9: IAS-9*, **9**, pp. 341–348.

[61] Sznaier, M., Cloutier, J., Hull, R., Jacques, D., and Mracek, C., 2000, "Receding horizon control Lyapunov function approach to suboptimal regulation of nonlinear systems," *Journal of Guidance, Control, and Dynamics*, **23**(3), pp. 399–405.

[62] Thrun, S., Burgard, W., and Fox, D., 2005, *Probability Robotics*, MIT Press, Cambridge, MA.

[63] Valdmanis, M., 2010, "Localization and navigation of a holonomic indoor airship using on-board sensors," Master's thesis, McGill University.

[64] Van Der Merwe, R., Wan, E., and Julier, S., 2004, "Sigma-point Kalman filters for nonlinear estimation and sensor-fusion: applications to integrated navigation," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Citeseer, p. 5120.

[65] Wan, E. and Van Der Merwe, R., 2000, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, IEEE, pp. 153–158.

[66] Yu, J., Jadbabaie, A., Primbs, J., and Huang, Y., 2001, "Comparison of nonlinear control design techniques on a model of the Caltech ducted fan," *Automatica*, **37**(12), pp. 1971–1978.

[67] Zanetti, R., Majji, M., Bishop, R., and Mortari, D., 2009, "Norm-Constrained Kalman Filtering," *Journal of Guidance, Control, and Dynamics*, **32**(5), pp. 1458–1465.

[68] Zhang, H. and Ostrowski, J., 1999, "Visual servoing with dynamics: control of an unmanned blimp," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, IEEE, **1**, pp. 618–623.

[69] Zufferey, J., Floreano, D., van Leeuwen, M., and Merenda, T., 2010, "Evolving vision-based flying robots," in *Biologically Motivated Computer Vision*, Springer, pp. 13–29.

[70] Zufferey, J., Guanella, A., Beyeler, A., and Floreano, D., 2006, "Flying over the reality gap: from simulated to real indoor airships," *Autonomous Robots*, **21**(3), pp. 243–254.

Appendix A

Jacobian Linearization of Airship Model

$$\mathbf{A}_0 = \begin{bmatrix} \mathbf{A}_q \\ \mathbf{A}_p \\ \mathbf{A}_\omega \\ \mathbf{A}_v \end{bmatrix} \tag{A.1}$$

where

$$\mathbf{A}_q^T = \begin{bmatrix} 0 & \frac{\omega_x}{2} + 2q_0q_2\omega_z - 2q_0q_3\omega_y & \frac{\omega_y}{2} - 2q_0q_1\omega_z + 2q_0q_3\omega_x & \frac{\omega_z}{2} + 2q_0q_1\omega_y - 2q_0q_2\omega_x \\ -\frac{\omega_x}{2} & 2q_1\left(q_2\omega_z - q_3\omega_y\right) & 2q_1q_3\omega_x - \omega_z\left(2q_1^2 + \frac{1}{2}\right) & \omega_y\left(2q_1^2 + \frac{1}{2}\right) - 2q_1q_2\omega_x \\ -\frac{\omega_y}{2} & \omega_z\left(2q_2^2 + \frac{1}{2}\right) - 2q_2q_3\omega_y & -2q_2\left(q_1\omega_z - q_3\omega_x\right) & 2q_1q_2\omega_y - \omega_x\left(2q_2^2 + \frac{1}{2}\right) \\ -\frac{\omega_z}{2} & 2q_2q_3\omega_z - \omega_y\left(2q_3^2 + \frac{1}{2}\right) & \omega_x\left(2q_3^2 + \frac{1}{2}\right) - 2q_1q_3\omega_z & 2q_3\left(q_1\omega_y - q_2\omega_x\right) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{q_1}{2} & \frac{q_0}{2} & \frac{q_3}{2} & -\frac{q_2}{2} \\ -\frac{q_2}{2} & -\frac{q_3}{2} & \frac{q_0}{2} & \frac{q_1}{2} \\ -\frac{q_3}{2} & \frac{q_2}{2} & -\frac{q_1}{2} & \frac{q_0}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.2}$$

$$\mathbf{A}_p^T = \begin{bmatrix} 2q_2v_z - 2q_3v_y & 2q_3v_x - 2q_1v_z & 2q_1v_y - 2q_2v_x \\ 2q_2v_y + 2q_3v_z & 2q_2v_x - 4q_1v_y - 2q_0v_z & 2q_0v_y - 4q_1v_z + 2q_3v_x \\ 2q_0v_z + 2q_1v_y - 4q_2v_x & 2q_1v_x + 2q_3v_z & 2q_3v_y - 4q_2v_z - 2q_0v_x \\ 2q_1v_z - 2q_0v_y - 4q_3v_x & 2q_0v_x + 2q_2v_z - 4q_3v_y & 2q_1v_x + 2q_2v_y \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -2q_2^2 - 2q_3^2 + 1 & 2q_0q_3 + 2q_1q_2 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & -2q_1^2 - 2q_3^2 + 1 & 2q_0q_1 + 2q_2q_3 \\ 2q_0q_2 + 2q_1q_3 & 2q2q_3 - 2q_0q_1 & -2q_1^2 - 2q_2^2 + 1 \end{bmatrix} \tag{A.3}$$

$$\mathbf{A}_\omega^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ a_{\omega 81} & a_{\omega 82} & a_{\omega 83} \\ a_{\omega 91} & a_{\omega 92} & a_{\omega 93} \\ a_{\omega 101} & a_{\omega 102} & a_{\omega 103} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.4}$$

In Eq. (A.4), we assume the inertia matrix of the airship to be:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{A.5}$$

$$\begin{aligned} a_{\omega 81} = \quad & (2\omega_x I_{xy}^2 I_{yz} - 2\omega_x I_{xy} I_{xz} I_{yy} + 2\omega_x I_{xy} I_{xz} I_{zz} + \omega_y I_{xy} I_{yy} I_{yz} - \omega_z I_{xy} I_{yy} I_{zz} \\ & + 2\omega_z I_{xy} I_{yz}^2 + \omega_y I_{xy} I_{yz} I_{zz} - I_{xx}\omega_y I_{xy} I_{yz} + \omega_z I_{xy} I_{zz}^2 - I_{xx}\omega_z I_{xy} I_{zz} - 2\omega_x I_{xz}^2 I_{yz} \\ & - \omega_y I_{xz} {I_{yy}}^2 - \omega_z I_{xz} I_{yy} I_{yz} + \omega_y I_{xz} I_{yy} I_{zz} + I_{xx}\omega_y I_{xz} I_{yy} - 2\omega_y I_{xz} {I_{yz}}^2 \\ & - \omega_z I_{xz} I_{yz} I_{zz} + I_{xx}\omega_z I_{xz} I_{yz}) / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - \\ & I_{xx} I_{yy} I_{zz}) \end{aligned}$$

$$\begin{aligned} a_{\omega 82} = \quad & -(-\omega_y {I_{xx}}^2 I_{yz} - \omega_z {I_{xx}}^2 I_{zz} + \omega_y I_{xx} I_{xy} I_{xz} + 2\omega_x I_{xx} I_{xy} I_{yz} \\ & + \omega_z I_{xx} {I_{xz}}^2 + 2\omega_x I_{xx} I_{xz} I_{zz} + \omega_z I_{xx} {I_{yz}}^2 + \omega_y I_{xx} I_{yz} I_{zz} + I_{yy}\omega_y I_{xx} I_{yz} \\ & + \omega_z I_{xx} {I_{zz}}^2 - 2\omega_x {I_{xy}}^2 I_{xz} - \omega_z {I_{xy}}^2 I_{zz} + \omega_y I_{xy} I_{xz} I_{zz} - I_{yy}\omega_y I_{xy} I_{xz} \\ & - 2\omega_x {I_{xz}}^3 - 2\omega_y {I_{xz}}^2 I_{yz} - \omega_z {I_{xz}}^2 I_{zz}) / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 \\ & + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}) \end{aligned}$$

$$\begin{aligned} a_{\omega 83} = \quad & (-\omega_y {I_{xx}}^2 I_{yy} - \omega_z {I_{xx}}^2 I_{yz} + \omega_y I_{xx} {I_{xy}}^2 + \omega_z I_{xx} I_{xy} I_{xz} + 2\omega_x I_{xx} I_{xy} I_{yy} \\ & + 2\omega_x I_{xx} I_{xz} I_{yz} + \omega_y I_{xx} {I_{yy}}^2 + \omega_z I_{xx} I_{yy} I_{yz} + \omega_y I_{xx} {I_{yz}}^2 + I_{zz}\omega_z I_{xx} I_{yz} - 2\omega_x {I_{xy}}^3 \\ & - \omega_y {I_{xy}}^2 I_{yy} - 2\omega_z {I_{xy}}^2 I_{yz} - 2\omega_x I_{xy} {I_{xz}}^2 + \omega_z I_{xy} I_{xz} I_{yy} - I_{zz}\omega_z I_{xy} I_{xz} - \omega_y {I_{xz}}^2 I_{yy}) \\ & / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}) \end{aligned}$$

$$
\begin{aligned}
a_{\omega 91} = \quad & (-2\omega_y {I_{xy}}^2 I_{yz} - \omega_z {I_{xy}}^2 I_{zz} + 2\omega_y I_{xy} I_{xz} I_{yy} + \omega_x I_{xy} I_{yy} I_{yz} + \omega_x I_{xy} I_{yz} I_{zz} \\
& - I_{xx}\omega_x I_{xy} I_{yz} + \omega_z {I_{xz}}^2 I_{yy} - \omega_x I_{xz} {I_{yy}}^2 + \omega_x I_{xz} I_{yy} I_{zz} + I_{xx}\omega_x I_{xz} I_{yy} - 2\omega_x I_{xz} {I_{yz}}^2 \\
& - \omega_z {I_{yy}}^2 I_{zz} + \omega_z I_{yy} {I_{yz}}^2 + 2\omega_y I_{yy} I_{yz} I_{zz} + \omega_z I_{yy} {I_{zz}}^2 - 2\omega_y {I_{yz}}^3 - \omega_z {I_{yz}}^2 I_{zz}) \\
& / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz})
\end{aligned}
$$

$$
\begin{aligned}
a_{\omega 92} = \quad & (\omega_x {I_{xx}}^2 I_{yz} - \omega_x I_{xx} I_{xy} I_{xz} + 2\omega_y I_{xx} I_{xy} I_{yz} + \omega_z I_{xx} I_{xy} I_{zz} + \\
& \omega_z I_{xx} I_{xz} I_{yz} - \omega_x I_{xx} I_{yz} I_{zz} - I_{yy}\omega_x I_{xx} I_{yz} - 2\omega_y {I_{xy}}^2 I_{xz} - 2\omega_z I_{xy} {I_{xz}}^2 - \omega_x I_{xy} I_{xz} I_{zz} \\
& + I_{yy}\omega_x I_{xy} I_{xz} - 2\omega_y I_{xy} I_{yz} I_{zz} - \omega_z I_{xy} {I_{zz}}^2 + I_{yy}\omega_z I_{xy} I_{zz} + 2\omega_x {I_{xz}}^2 I_{yz} + 2\omega_y I_{xz} {I_{yz}}^2 \\
& + \omega_z I_{xz} I_{yz} I_{zz} - I_{yy}\omega_z I_{xz} I_{yz}) / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz})
\end{aligned}
$$

$$
\begin{aligned}
a_{\omega 93} = \quad & (\omega_x {I_{xx}}^2 I_{yy} - \omega_x I_{xx} {I_{xy}}^2 + 2\omega_y I_{xx} I_{xy} I_{yy} + \omega_z I_{xx} I_{xy} I_{yz} + \omega_z I_{xx} I_{xz} I_{yy} \\
& - \omega_x I_{xx} {I_{yy}}^2 - \omega_x I_{xx} {I_{yz}}^2 - 2\omega_y {I_{xy}}^3 - 2\omega_z {I_{xy}}^2 I_{xz} + \omega_x {I_{xy}}^2 I_{yy} + \omega_z I_{xy} I_{yy} I_{yz} \\
& - 2\omega_y I_{xy} {I_{yz}}^2 - I_{zz}\omega_z I_{xy} I_{yz} + \omega_x {I_{xz}}^2 I_{yy} - \omega_z I_{xz} {I_{yy}}^2 + 2\omega_y I_{xz} I_{yy} I_{yz} + I_{zz}\omega_z I_{xz} I_{yy}) \\
& / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz})
\end{aligned}
$$

$$
\begin{aligned}
a_{\omega 101} = \quad & -(\omega_y {I_{xy}}^2 I_{zz} + 2\omega_z I_{xy} I_{xz} I_{zz} + \omega_x I_{xy} I_{yy} I_{zz} - 2\omega_x I_{xy} {I_{yz}}^2 - \omega_x I_{xy} {I_{zz}}^2 \\
& + I_{xx}\omega_x I_{xy} I_{zz} - \omega_y {I_{xz}}^2 I_{yy} - 2\omega_z {I_{xz}}^2 I_{yz} + \omega_x I_{xz} I_{yy} I_{yz} + \omega_x I_{xz} I_{yz} I_{zz} - I_{xx}\omega_x I_{xz} I_{yz} \\
& + \omega_y {I_{yy}}^2 I_{zz} - \omega_y I_{yy} {I_{yz}}^2 + 2\omega_z I_{yy} I_{yz} I_{zz} - \omega_y I_{yy} {I_{zz}}^2 - 2\omega_z {I_{yz}}^3 + \omega_y {I_{yz}}^2 I_{zz}) \\
& / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz})
\end{aligned}
$$

$$
\begin{aligned}
a_{\omega 102} = \quad & (\omega_x {I_{xx}}^2 I_{zz} + \omega_y I_{xx} I_{xy} I_{zz} - \omega_x I_{xx} {I_{xz}}^2 + \omega_y I_{xx} I_{xz} I_{yz} + 2\omega_z I_{xx} I_{xz} I_{zz} \\
& - \omega_x I_{xx} {I_{yz}}^2 - \omega_x I_{xx} {I_{zz}}^2 + \omega_x {I_{xy}}^2 I_{zz} - 2\omega_y I_{xy} {I_{xz}}^2 + 2\omega_z I_{xy} I_{yz} I_{zz} - \omega_y I_{xy} {I_{zz}}^2 \\
& + I_{yy} \omega_y I_{xy} I_{zz} - 2\omega_z {I_{xz}}^3 + \omega_x {I_{xz}}^2 I_{zz} - 2\omega_z I_{xz} {I_{yz}}^2 + \omega_y I_{xz} I_{yz} I_{zz} - I_{yy} \omega_y I_{xz} I_{yz}) \\
& / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz})
\end{aligned}
$$

$$
\begin{aligned}
a_{\omega 103} = \quad & -(\omega_x {I_{xx}}^2 I_{yz} - \omega_x I_{xx} I_{xy} I_{xz} + \omega_y I_{xx} I_{xy} I_{yz} + \omega_y I_{xx} I_{xz} I_{yy} + 2\omega_z I_{xx} I_{xz} I_{yz} \\
& - \omega_x I_{xx} I_{yy} I_{yz} - I_{zz} \omega_x I_{xx} I_{yz} - 2\omega_y {I_{xy}}^2 I_{xz} + 2\omega_x {I_{xy}}^2 I_{yz} - 2\omega_z I_{xy} {I_{xz}}^2 - \omega_x I_{xy} I_{xz} I_{yy} \\
& + I_{zz} \omega_x I_{xy} I_{xz} + \omega_y I_{xy} I_{yy} I_{yz} + 2\omega_z I_{xy} {I_{yz}}^2 - I_{zz} \omega_y I_{xy} I_{yz} - \omega_y I_{xz} {I_{yy}}^2 - 2\omega_z I_{xz} I_{yy} I_{yz} \\
& + I_{zz} \omega_y I_{xz} I_{yy}) / (I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz})
\end{aligned}
$$

$$
\mathbf{A}_\omega^T = \mathbf{0}_{13\times 3} \tag{A.6}
$$

$$
\mathbf{B}_0 = \begin{bmatrix} \mathbf{0}_{7\times 6} \\ \mathbf{b}_{t1} \\ \mathbf{b}_{t2} \\ \mathbf{b}_{t3} \\ \mathbf{B}_f \end{bmatrix} \tag{A.7}
$$

where,

$$
\mathbf{b}_{t1}^T = \begin{bmatrix}
\frac{L(I_{xz} I_{yz} - I_{xy} I_{zz})}{I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}} \\
-\frac{L({I_{yz}}^2 - I_{yy} I_{zz})}{I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}} \\
\frac{L(I_{xy} I_{yz} - I_{xz} I_{yy})}{I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}} \\
\frac{L({I_{yz}}^2 - I_{yy} I_{zz})}{I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}} \\
-\frac{L(I_{xy} I_{yz} - I_{xz} I_{yy})}{I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}} \\
-\frac{L(I_{xz} I_{yz} - I_{xy} I_{zz})}{I_{zz} {I_{xy}}^2 - 2 I_{xy} I_{xz} I_{yz} + I_{yy} {I_{xz}}^2 + I_{xx} {I_{yz}}^2 - I_{xx} I_{yy} I_{zz}}
\end{bmatrix} \tag{A.8}
$$

$$\mathbf{b}_{t2}^{T} = \begin{bmatrix} -\frac{L\left(I_{xz}{}^2 - I_{xx}I_{zz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ \frac{L\left(I_{xz}I_{yz} - I_{xy}I_{zz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ \frac{L\left(I_{xy}I_{xz} - I_{xx}I_{yz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ -\frac{L\left(I_{xz}I_{yz} - I_{xy}I_{zz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ -\frac{L\left(I_{xy}I_{xz} - I_{xx}I_{yz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ \frac{L\left(I_{xz}{}^2 - I_{xx}I_{zz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \end{bmatrix} \tag{A.9}$$

$$\mathbf{b}_{t3}^{T} = \begin{bmatrix} \frac{L\left(I_{xy}I_{xz} - I_{xx}I_{yz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ \frac{L\left(I_{xy}I_{yz} - I_{xz}I_{yy}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ -\frac{L\left(I_{xy}{}^2 - I_{xx}I_{yy}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ -\frac{L\left(I_{xy}I_{yz} - I_{xz}I_{yy}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ \frac{L\left(I_{xy}{}^2 - I_{xx}I_{yy}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \\ -\frac{L\left(I_{xy}I_{xz} - I_{xx}I_{yz}\right)}{I_{zz}I_{xy}{}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 - I_{xx}I_{yy}I_{zz}} \end{bmatrix} \tag{A.10}$$

$$\mathbf{B}_f = \begin{bmatrix} 0 & 0 & \frac{1}{m_a + m_{add}} & 0 & \frac{1}{m_a + m_{add}} & 0 \\ 0 & \frac{1}{m_a + m_{add}} & 0 & \frac{1}{m_a + m_{add}} & 0 & 0 \\ \frac{1}{m_a + m_{add}} & 0 & 0 & 0 & 0 & \frac{1}{m_a + m_{add}} \end{bmatrix} \tag{A.11}$$

Appendix B

Extended Linearization of Airship Model

$$\mathbf{A}(\mathbf{x}) =$$

$$\begin{pmatrix}
0 & \cdots & 0 & -\frac{q_1}{2} & -\frac{q_2}{2} & -\frac{q_3}{2} & 0 & 0 & 0 \\
0 & \cdots & 0 & \frac{q_0}{2} & -\frac{q_3}{2} & \frac{q_2}{2} & 0 & 0 & 0 \\
0 & \cdots & 0 & \frac{q_3}{2} & \frac{q_0}{2} & -\frac{q_1}{2} & 0 & 0 & 0 \\
0 & \cdots & 0 & -\frac{q_2}{2} & \frac{q_1}{2} & \frac{q_0}{2} & 0 & 0 & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & -2q_2^2 - 2q_3^2 + 1 & 2q_1q_2 - 2q_0q_3 & 2q_0q_2 + 2q_1q_3 \\
0 & \cdots & 0 & 0 & 0 & 0 & 2q_0q_3 + 2q_1q_2 & -2q_1^2 - 2q_3^2 + 1 & 2q_2q_3 - 2q_0q_1 \\
0 & \cdots & 0 & 0 & 0 & 0 & 2q_1q_3 - 2q_0q_2 & 2q_0q_1 + 2q_2q_3 & -2q_1^2 - 2q_2^2 + 1 \\
0 & \cdots & 0 & \frac{\omega_{xx}}{I_0} & \frac{\omega_{xy}}{I_0} & \frac{\omega_{xz}}{I_0} & 0 & 0 & 0 \\
0 & \cdots & 0 & \frac{\omega_{yx}}{I_0} & \frac{\omega_{yy}}{I_0} & \frac{\omega_{yz}}{I_0} & 0 & 0 & 0 \\
0 & \cdots & 0 & \frac{\omega_{zx}}{I_0} & \frac{\omega_{zy}}{I_0} & \frac{\omega_{zz}}{I_0} & 0 & 0 & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \tag{B.1}$$

$$I_0 = I_{zz}I_{xy}^2 - 2I_{xy}I_{xz}I_{yz} + I_{yy}I_{xz}^2 + I_{xx}I_{yz}^2 - I_{xx}I_{yy}I_{zz} \tag{B.2}$$

$$\begin{aligned}\omega_{xx} = (&I_{xy}^2I_{yz} - I_{xz}^2I_{yz} - I_{xy}I_{xz}I_{yy} + I_{xy}I_{xz}I_{zz})\omega_x + \alpha(-I_{xz}I_{yy}^2 - 2I_{xz}I_{yz}^2 \\ &- I_{xx}I_{xy}I_{yz} + I_{xx}I_{xz}I_{yy} + I_{xy}I_{yy}I_{yz} + I_{xy}I_{yz}I_{zz} + I_{xz}I_{yy}I_{zz})\omega_y + \beta(2I_{xy}I_{yz}^2 \\ &+ I_{xy}I_{zz}^2 + I_{xx}I_{xz}I_{yz} - I_{xx}I_{xy}I_{zz} - I_{xz}I_{yy}I_{yz} - I_{xy}I_{yy}I_{zz} - I_{xz}I_{yz}I_{zz})\omega_z\end{aligned} \tag{B.3}$$

$$\begin{aligned}\omega_{xy} = (1-\alpha)(&-I_{xz}I_{yy}^2 - 2I_{xz}I_{yz}^2 - I_{xx}I_{xy}I_{yz} + I_{xx}I_{xz}I_{yy} + I_{xy}I_{yy}I_{yz} + I_{xy}I_{yz}I_{zz} \\ &+ I_{xz}I_{yy}I_{zz})\omega_x + (I_{yy}I_{yz}I_{zz} + I_{xy}I_{xz}I_{yy} - I_{yz}^3 - I_{xy}^2I_{yz})\omega_y + \gamma(I_{xz}^2I_{yy} + I_{yy}I_{yz}^2 \\ &- I_{xy}^2I_{zz} + I_{yy}I_{zz}^2 - I_{yy}^2I_{zz} - I_{yz}^2I_{zz})\omega_z\end{aligned} \tag{B.4}$$

$$\begin{aligned}\omega_{xz} = (1-\beta)(&2I_{xy}I_{yz}^2 + I_{xy}I_{zz}^2 + I_{xx}I_{xz}I_{yz} - I_{xx}I_{xy}I_{zz} - I_{xz}I_{yy}I_{yz} - I_{xy}I_{yy}I_{zz} \\ &- I_{xz}I_{yz}I_{zz})\omega_x + (1-\gamma)(I_{xz}^2I_{yy} + I_{yy}I_{yz}^2 - I_{xy}^2I_{zz} + I_{yy}I_{zz}^2 - I_{yy}^2I_{zz} \\ &- I_{yz}^2I_{zz})\omega_y + (I_{yz}^3 + I_{xz}^2I_{yz} - I_{xy}I_{xz}I_{zz} - I_{yy}I_{yz}I_{zz})\omega_z\end{aligned} \tag{B.5}$$

$$\begin{aligned}\omega_{yx} = \omega_x(&I_{xy}{}^2I_{xz} - I_{xx}I_{yz}I_{xy} + I_{xz}{}^3 - I_{xx}I_{zz}I_{xz}) - \alpha\omega_y(I_{xx}I_{xy}I_{xz} - 2I_{xz}{}^2I_{yz} \\ &- I_{xx}{}^2I_{yz} - I_{xy}I_{xz}I_{yy} + I_{xx}I_{yy}I_{yz} + I_{xy}I_{xz}I_{zz} + I_{xx}I_{yz}I_{zz}) - \beta\omega_z(-I_{xx}{}^2I_{zz} \\ &+ I_{xx}I_{xz}{}^2 + I_{xx}I_{yz}{}^2 + I_{xx}I_{zz}{}^2 - I_{xy}{}^2I_{zz} - I_{xz}{}^2I_{zz})\end{aligned} \tag{B.6}$$

$$\begin{aligned}\omega_{yy} = \gamma\omega_z(&I_{xx}I_{xz}I_{yz} - I_{xy}I_{zz}{}^2 - 2I_{xy}I_{xz}{}^2 + I_{xx}I_{xy}I_{zz} - I_{xz}I_{yy}I_{yz} + I_{xy}I_{yy}I_{zz} \\ &+ I_{xz}I_{yz}I_{zz}) - \omega_y(I_{xy}{}^2I_{xz} - I_{xz}I_{yz}{}^2 - I_{xx}I_{xy}I_{yz} + I_{xy}I_{yz}I_{zz}) + \omega_x(\alpha-1)(I_{xx}I_{xy}I_{xz} \\ &- 2I_{xz}{}^2I_{yz} - I_{xx}{}^2I_{yz} - I_{xy}I_{xz}I_{yy} + I_{xx}I_{yy}I_{yz} + I_{xy}I_{xz}I_{zz} + I_{xx}I_{yz}I_{zz})\end{aligned} \tag{B.7}$$

$$\omega_{yz} = \omega_x(\beta - 1)(-{I_{xx}}^2 I_{zz} + I_{xx}{I_{xz}}^2 + I_{xx}{I_{yz}}^2 + I_{xx}{I_{zz}}^2 - {I_{xy}}^2 I_{zz} - {I_{xz}}^2 I_{zz})$$
$$- \omega_y(\gamma - 1)(I_{xx}I_{xz}I_{yz} - I_{xy}{I_{zz}}^2 - 2I_{xy}{I_{xz}}^2 + I_{xx}I_{xy}I_{zz} - I_{xz}I_{yy}I_{yz} + I_{xy}I_{yy}I_{zz}$$
$$+ I_{xz}I_{yz}I_{zz}) - \omega_z({I_{xz}}^3 + I_{xz}{I_{yz}}^2 - I_{xx}I_{zz}I_{xz} - I_{xy}I_{zz}I_{yz}) \quad \text{(B.8)}$$

$$\omega_{zx} = \beta\omega_z(I_{xx}I_{xy}I_{xz} - 2{I_{xy}}^2 I_{yz} - {I_{xx}}^2 I_{yz} + I_{xy}I_{xz}I_{yy} + I_{xx}I_{yy}I_{yz} - I_{xy}I_{xz}I_{zz}$$
$$+ I_{xx}I_{yz}I_{zz}) - \omega_x({I_{xy}}^3 + I_{xy}{I_{xz}}^2 - I_{xx}I_{yy}I_{xy} - I_{xx}I_{yz}I_{xz}) + \alpha\omega_y(-{I_{xx}}^2 I_{yy}$$
$$+ I_{xx}{I_{xy}}^2 + I_{xx}{I_{yy}}^2 + I_{xx}{I_{yz}}^2 - {I_{xy}}^2 I_{yy} - {I_{xz}}^2 I_{yy}) \quad \text{(B.9)}$$

$$\omega_{zy} = \omega_y({I_{xy}}^3 + I_{xy}{I_{yz}}^2 - I_{xx}I_{yy}I_{xy} - I_{xz}I_{yy}I_{yz}) - \gamma\omega_z(I_{xx}I_{xy}I_{yz} - I_{xz}{I_{yy}}^2$$
$$- 2{I_{xy}}^2 I_{xz} + I_{xx}I_{xz}I_{yy} + I_{xy}I_{yy}I_{yz} - I_{xy}I_{yz}I_{zz} + I_{xz}I_{yy}I_{zz}) - \omega_x(\alpha - 1)(-{I_{xx}}^2 I_{yy}$$
$$+ I_{xx}{I_{xy}}^2 + I_{xx}{I_{yy}}^2 + I_{xx}{I_{yz}}^2 - {I_{xy}}^2 I_{yy} - {I_{xz}}^2 I_{yy}) \quad \text{(B.10)}$$

$$\omega_{zz} = \omega_z(I_{xy}{I_{xz}}^2 - I_{xy}{I_{yz}}^2 - I_{xx}I_{xz}I_{yz} + I_{xz}I_{yy}I_{yz}) - \omega_x(\beta - 1)(I_{xx}I_{xy}I_{xz}$$
$$- 2{I_{xy}}^2 I_{yz} - {I_{xx}}^2 I_{yz} + I_{xy}I_{xz}I_{yy} + I_{xx}I_{yy}I_{yz} - I_{xy}I_{xz}I_{zz} + I_{xx}I_{yz}I_{zz}) + \omega_y(\gamma - 1)$$
$$(I_{xx}I_{xy}I_{yz} - I_{xz}{I_{yy}}^2 - 2{I_{xy}}^2 I_{xz} + I_{xx}I_{xz}I_{yy} + I_{xy}I_{yy}I_{yz} - I_{xy}I_{yz}I_{zz} + I_{xz}I_{yy}I_{zz}) \quad \text{(B.11)}$$

Appendix C

Conjugate and product of quaternions

Assume two quaternions as

$$\mathbf{q}_1 = \begin{bmatrix} q_1 \\ \mathbf{q}_{v,1} \end{bmatrix} \qquad \mathbf{q}_2 = \begin{bmatrix} q_2 \\ \mathbf{q}_{v,2} \end{bmatrix}$$

The conjugate of a quaternion is computed by

$$\mathbf{q}_1' = \begin{bmatrix} q_1 \\ -\mathbf{q}_{v,1} \end{bmatrix} \tag{C.1}$$

The quaternion product is then presented by

$$\tilde{\mathbf{q}} = \mathbf{q}_1' * \mathbf{q}_2 = \begin{bmatrix} q_1 q_2 + \mathbf{q}_{v,1} \cdot \mathbf{q}_{v,2} \\ q_1 \mathbf{q}_{v,2} - q_2 \mathbf{q}_{v,1} - \mathbf{q}_{v,1} \times \mathbf{q}_{v,2} \end{bmatrix} \tag{C.2}$$

Appendix D

Model of Airship's Power Consumption

For the sake of controller evaluation, the relationship between the power consumed by the airship and the control inputs sent by the control station needs to be studied. The thrusters on-board are brushed DC motors driven by the PWM signal, as shown in Fig. D.1. In the diagram, V_s is the power source; R_C is the resistance of the motor driving cable (the cable resistance is not trivial considering the average length of 1.5 m); R_M is the armature resistance and L_M is the armature inductance. As the DC motor rotates, it produces CEMF (counter/back electromotive force) as indicated in the circuit diagram.

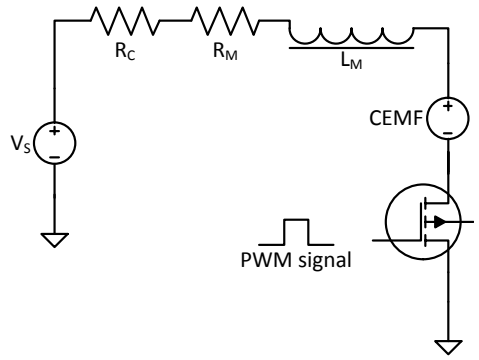


Figure D.1 Brushed DC motor on-board with PWM control

According to the principle of PWM control, the average voltage supplied by the DC

source V_S is given by:

$$V = \frac{1}{T}\left(\int_0^D V_{ON}dt + \int_D^T V_{OFF}dt\right) = \frac{D}{T}V_S \tag{D.1}$$

where T is the modulating period; V_{ON} and V_{OFF} are the voltage outputs when the MOSFET is on and off; D is the command sent by the control station. Ideally, the driving voltage is proportional to the controller command. Ignoring the energy losses of the MOSFET, the power consumed by each propeller is:

$$P = VI \tag{D.2}$$

where I is the armature current. For a DC motor,

$$V - V_E = L_M\frac{dI}{dt} + (R_M + R_C)I \tag{D.3}$$

where V_E is the CEMF voltage. In steady state, we have:

$$I = \frac{V - V_E}{R_M + R_C} = \frac{T}{K_T} \tag{D.4}$$

where K_T is the torque constant of the motor and T is the torque applied to the motor shaft. The value of T depends on many factors such as the rotor speed, the blade design, speed of the incoming air flow, etc. We assume that all thrusters on the airship are identical in construction and the incoming air flow is negligible compared with the accelerated air flow, since the airship's motion is very slow. The relationship between rotor speed Ω and motor torque T can be linearized in a wide operating range by using a constant K_ω as [1].

$$T = K_\omega\Omega \tag{D.5}$$

Therefore,

$$K_\omega \Omega = K_T I = K_T \left(\frac{V - V_E}{R_M + R_C} \right) \tag{D.6}$$

Recalling the principle of DC actuator, we obtain:

$$V_E = K_E \Omega \tag{D.7}$$

where K_E is the speed constant of the DC motor. Combining Eq. (D.6) and Eq. (D.7), we have:

$$\Omega = \frac{K_T}{K_\omega (R_M + R_C) + K_T K_E} V \tag{D.8}$$

Recalling Eq. (D.2), for each actuator, we have:

$$\begin{aligned} P &= V \frac{V - V_E}{R_M + R_C} \\ &= V \frac{V - K_E \Omega}{R_M + R_C} \\ &= \frac{K_\omega}{K_\omega (R_M + R_C) + K_T K_E} V^2 \end{aligned} \tag{D.9}$$

Therefore, the total power consumed by the actuating system is proportional to the square of the average voltage supplied, and hence, according to Eq. (D.1), it is proportional to the square of control inputs.

Appendix E

Measurement Noise Covariance Matrix for Vicon/IMU Integration

The following measurement noise covariance **R** is used for Vicon/IMU integration:

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7
2.17E-06	-3.14E-07	-1.33E-07	-3.82E-09	-5.86E-07	1.44E-07	-1.81E-06
-3.14E-07	1.64E-06	-1.96E-07	2.18E-07	-1.93E-07	8.96E-08	7.36E-07
-1.33E-07	-1.96E-07	2.28E-06	-4.20E-07	1.22E-06	-8.89E-08	-2.87E-07
-3.82E-09	2.18E-07	-4.20E-07	1.87E-06	-5.18E-07	-4.10E-07	5.49E-07
-5.86E-07	-1.93E-07	1.22E-06	-5.18E-07	4.12E-06	5.64E-07	-5.82E-08
1.44E-07	8.96E-08	-8.89E-08	-4.10E-07	5.64E-07	4.24E-06	-5.08E-07
-1.81E-06	7.36E-07	-2.87E-07	5.49E-07	-5.82E-08	-5.08E-07	4.62E-06
4.29E-07	6.00E-07	-3.17E-07	-1.00E-07	-7.23E-07	3.78E-07	-1.15E-06
8.92E-07	-6.27E-08	-1.21E-06	3.51E-07	-2.24E-07	2.21E-07	-1.09E-07
8.97E-07	-2.48E-07	-1.03E-06	2.12E-07	-8.79E-07	9.90E-07	-3.94E-07
-2.71E-06	-1.17E-06	5.82E-06	-3.26E-06	1.65E-05	1.07E-06	-1.12E-06
4.96E-07	-4.89E-07	-7.79E-07	-1.69E-06	1.28E-06	1.22E-05	-2.41E-06
-7.18E-06	4.91E-06	-1.54E-06	2.69E-06	1.85E-06	-9.32E-07	1.86E-05

Column 8	Column 9	Column 10	Column 11	Column 12	Column 13
4.29E-07	8.92E-07	8.97E-07	-2.71E-06	4.96E-07	-7.18E-06
6.00E-07	-6.27E-08	-2.48E-07	-1.17E-06	-4.89E-07	4.91E-06
-3.17E-07	-1.21E-06	-1.03E-06	5.82E-06	-7.79E-07	-1.54E-06
-1.00E-07	3.51E-07	2.12E-07	-3.26E-06	-1.69E-06	2.69E-06
-7.23E-07	-2.24E-07	-8.79E-07	1.65E-05	1.28E-06	1.85E-06
3.78E-07	2.21E-07	9.90E-07	1.07E-06	1.22E-05	-9.32E-07
-1.15E-06	-1.09E-07	-3.94E-07	-1.12E-06	-2.41E-06	1.86E-05
1.25E-03	5.80E-04	7.89E-04	3.69E-06	-6.84E-07	2.63E-06
5.80E-04	2.04E-03	1.00E-03	-7.12E-06	-1.75E-06	8.16E-06
7.89E-04	1.00E-03	2.03E-03	-1.44E-05	-1.53E-05	4.63E-06
3.69E-06	-7.12E-06	-1.44E-05	8.47E-04	8.52E-05	4.59E-05
-6.84E-07	-1.75E-06	-1.53E-05	8.52E-05	5.91E-04	-1.25E-04
2.63E-06	8.16E-06	4.63E-06	4.59E-05	-1.25E-04	9.94E-04