RESEARCH & DEVELOPMENT OF Q-BALLER – A SPHERICAL WHEELED ROBOT

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

JIAMIN WANG

Dr. Yuyi Lin, Thesis Supervisor

MAY 2017

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

RESEARCH & DEVELOPMENT OF Q-BALLER
– A SPHERICAL WHEELED ROBOT

presented by Jiamin Wang,

a candidate for the degree of Master of Science,

and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Yuyi Lin

Professor Ming Xin

Professor Roger Fales

Professor Stephen Montgomery-Smith

DEDICATION

This thesis is dedicated to my parents. Words cannot express the depth of my gratefulness to their love and support that has illuminated the path on my odyssey of pursuing the knowledge and realizing my dream. I could not have accomplished the research works without their inspiration and encouragement.

ACKNOWLEDGEMENTS

I would like to extend my appreciation to all the people who helped me with my research. I would like express my sincere gratitude to Dr. Yuyi Lin for offering me the interesting and meaningful research opportunity, the teaching in mechanical engineering and the supports in both my research work and life. It would also be an honor for me to extend my gratefulness to Dr. Ming Xin, Dr. Roger Fales and Dr. Stephen Montgomery-Smith for their guidance and enlightenment in dynamics, control and applied mathematics.

I would like to thank my friends Mr. Yueqi Yu, Mr. Ming Du and Mr. Xingfang Yuan for their help and assistance with electronic engineering, embedded system software development and artificial intelligence. I would also like to thank my colleagues Dr. Qingbing Tong, Dr. Zhengwei Nie, Mr. Fangrui Ding and Mr. Cecil Shy for their support and guidance in my research.

Last but not least, I would like to express my gratitude to my families and friends who have supported not only my research work but also my daily life. It has been my great honor to share with you this fruitful, memorable and meaningful journey.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF SIMULATIONS & EXPERIMENTS

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS & TERMINOLOGIES

Q-Baller: The Spherical Wheeled Robot Introduced in this Thesis.

BDC: Brushed Direct Current (Motor).

BLDC: Brushless Direct Current (Motor).

Frame G: Ground Coordinate Frame (the absolute frame based on the environment).

Frame O: Orientation Coordinate Frame (a local frame indicating the yawing of model).

Frame L: Local Coordinate Frame (a local frame indicating all attitudes of model).

Zero Point: The point at which all states (positions and velocities) are zero (for Q-Baller).

LQR: Linear Quadratic Regulator.

Spherical Simplex: N-D Simplex on a (N+1)-D Sphere Surface (spherical triangle, etc.)

CGS: Continuous Gain Scheduling (a method introduced in Chapter 5).

EL-CGS: Energy Leveling - Continuous Gain Scheduling (introduced in Chapter 5).

HMI: Human-Machine Interface.

JRB: JMechW Robotic Board (a robotic circuit board introduced in Chapter 7).

PCB: Printed Circuit Board.

JY901: Motion Sensor (integration of accelerometer, gyroscopic and compass sensors).

STM32: STM32F407VET6 (a microprocessor by STMicroelectronics).

GPIO: General Purpose Input/Output (a utility of microprocessor).

TIM: Timer (a utility of microprocessor used to count impulse and record time)

PWM: Pulse Width Modulation (a utility supported by TIM).

USART: Universal Synchronous/Asynchronous Receiver/Transmitter.

Interruption: Event-triggered functions that has the priority to interrupt other functions.

ABSTRACT

The Spherical Wheeled Robot (Ball-Bot) is a family of robots that can maintain balance standing on a ball and use it as its wheel to move around. In recent years, there have been several successful Ball-Bot designs.

We attempt to develop a new spherical wheeled robot product named “Q-Baller” to study its dynamics and control system. The Q-Baller has been designed to achieve the economic and effective prototyping. A detailed dynamic model of the mechatronic system has been established and analyzed. Control studies have been conducted based on the dynamic models, and new control methods has been proposed to realize continuous gain scheduling. Exclusive simulations have been performed to test the performance of the controllers and reference planning. The Q-Baller hardware has been prototyped and functional. Robotic circuit board, human machine interface and embedded control system have also been developed to make up the full robotic system.

The Q-Baller prototype will be tested after the system is fully adjusted, and further researches in control and robotics will be conducted in the future.

CHAPTER 1. INTRODUCTION OF RESEARCH

1.1. Spherical Wheeled Robot

Spherical Wheeled Robot, also known as Ball-Bot, is the kind of robot that can maintain balance standing on and moving around with a ball. Since 2005 when Carnegie Mellon University (CMU) accomplished the first Spherical Wheeled Robot, many successful Ball-Bot designs have been developed, a few are shown in Figure 1.1.

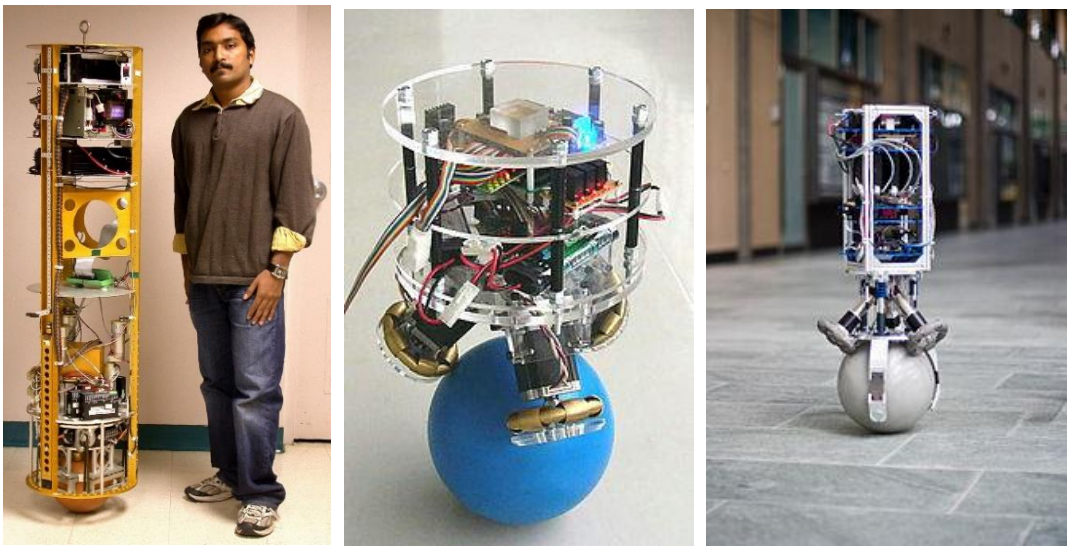


Figure 1.1: Successful Ball-Bot Designs

(Left: The CMU Ball-Bot [1]; Middle: The BallIP [2]; Right: The Rezero [3])

Even when Ball-Bots have similar features and ideas, the structures and the components used by the Ball-Bots can be very different. The CMU Ball-Bot used friction rollers to drive the balls. After the invention and popularization of Omni-Wheels [4], most Ball-Bots we see are now using Omni-Wheels in their friction drives. The BallIP Ball-Bot used 3 Omni-Wheel friction drives each driven by a Stepper Motor respectively. The

Rezero Ball-Bot, which is probably the most omnipotent Ball-Bot ever created, used Brushless DC Motors in their friction drivers.

Ball-Bots are not only interesting to design, making them maintain upright attitude and move around has been a dynamics and control problem worth studying. The control system of a Ball-Bot is like that of a 2-Dimensional Inverted Pendulum. The complex structure of the Ball-Bot will lead to highly coupled and nonlinear dynamics, making the control engineering very challenging.

1.2. Background & Motivation

Figure 1.2: Wheel Innovations

(Left: Mecanum Wheel; Right: Omni-Wheel)

The Ball-Bot design challenges the traditional concept of wheels and provides possibility for future mechatronic and robotic application. In traditional mindset, wheels are commonly depicted as a column installed on a shaft to facilitate the translation of kinetic momentum. The original wheel design has been innovated in the past 30 years for various newly emerged application purposes. For example, the Macanum Wheels [5] from

Fig 1.2 are designed for various robotic vehicles [6] to travel to any direction with different combination of wheel positioning and speed. Omni-Wheels, as mentioned before, has rings on its peripheral to neglect axial translational movement on the contact surface.

Recently, using sphere as wheel has been a popular idea. With a proper input, a sphere can be controlled to roll to any direction on a surface. However, to use sphere as wheels, one must deal with the following two common challenges:

1) On theory, the Ball-Bot's spherical wheel only has one contact point with the ground, making it inherently unstable when the center of gravity of the whole system is above the center of the spherical wheel.
2) The transmission of input energy need to be realized without using a shaft.
3) Therefore, currently the spherical wheels do not have much advantage in practical application.

Currently the most popular way to drive a spherical wheel is through friction drivers. The Ball-Bot research search for a better way to drive the spherical wheel. The research also intends to maximize the potential of spherical wheel by showing that the autonomous vehicle can be designed to move around with a single spherical wheel – only one contact point with the ground.

1.3. Objectives of Research & Challenges

With the development of mechatronic and computer technology, to develop a Ball-Bot with basic functionalities no longer requires expensive processor and high-end equipment. Therefore, we decide to develop our own Ball-Bot. The main objectives for the research project are listed below:

1) Reasonably design the Mechatronic System of Ball-Bot for an economic but effective prototype.
2) Study the Dynamics and Control System of Ball-Bot based on the designed model through simulation.
3) Experiment with the prototype (if successfully produced) to test the theory and conclusion reached from the previous two objectives.

The challenges of the project come not only from the research problem, but also from the limitation of research resource and engineering work load. Ball-Bot research project has limited funding, while as an individual project, the researcher need to work on all aspects of the product development.

CHAPTER 2. Q-BALLER – THE BALL-BOT DESIGN

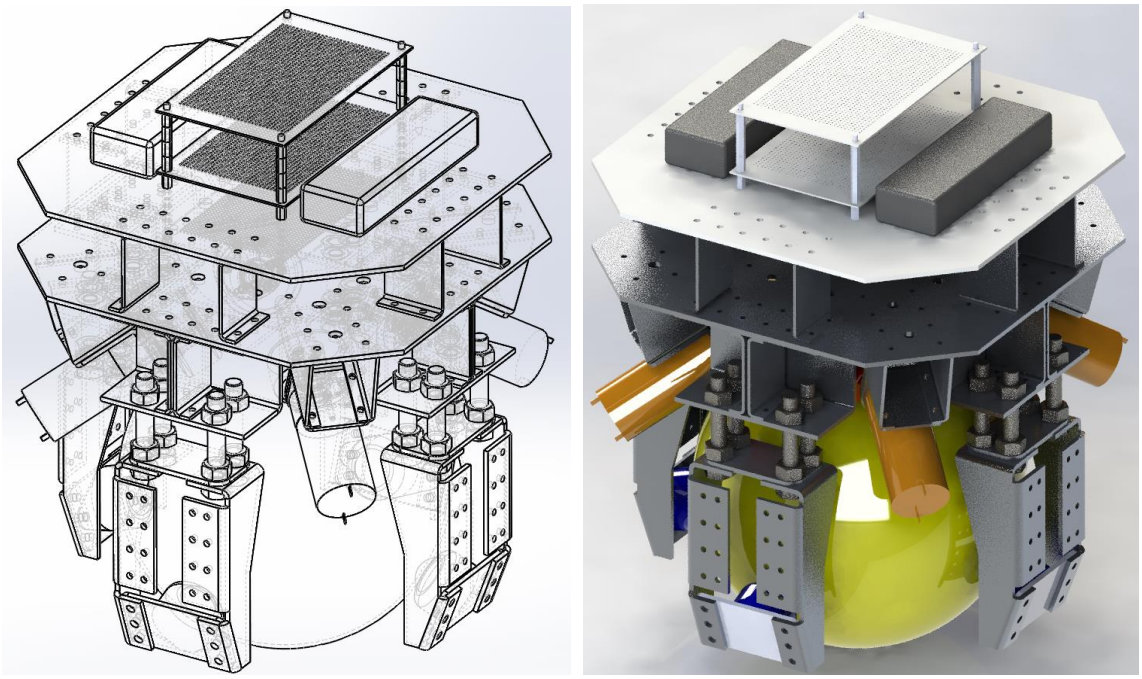


Figure 2.1: 3D Model of the Q-Baller Design

Presented in Fig. 2.1 is our Ball-Bot design nicknamed “Q-Baller”. The name was given based on the feature that the Ball-Bot was designed with 4 friction drive systems. The mechanical design of Q-Baller has gone through many considerations. The design process is introduced in the following sections.

2.1. Mechanical System Design of Q-Baller

The Q-Baller’s design was not simple. To design the Ball-Bot that can be prototyped with a relatively low cost while still to achieve basic functionalities and performances, we have considered design aspects including size, material, manufacture method, component selections and position arrangements.

2.1.1. Sizing & Mechanism Layout

The first step of the design procedure is to plan for the size and mechanism layout of the Ball-Bot based on the selection of the components. We planned to use four friction drive systems symmetrically allocated at the top of the sphere. The angle between two of the diagonally located friction wheels is 60 degrees, as presented in Fig. 2.2. Through such design we desire to realize symmetrical input from the control system. It only needs three friction drivers to realize all the basic control functions of a Ball-Bot, since there are only three degrees of freedom of the spherical wheel needed to be control.

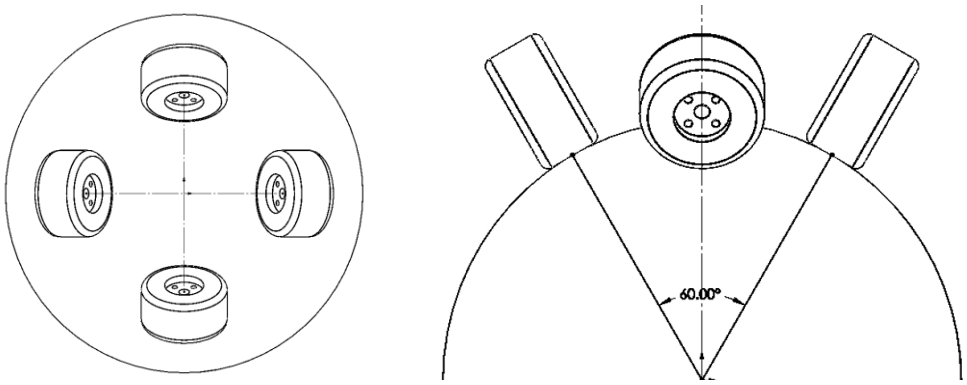


Figure 2.2: Position Arrangement of the Friction Wheels

Since the contact points of the friction wheels are close to each other, to ensure the contact stability we also applied bearing systems, which help to clamp the spherical wheel to the four friction drives.

The friction wheels we select for the friction drive systems are Omni-Wheels, since they are popular and can be purchased at relatively low cost. A high-quality Omni-Wheel with a large size are expensive. The final Omni-Wheel selection has a diameter of 48 mm and a maximum operational load of 3 Kg. According to previously determined wheel positions, the total load exerted on the omni-wheels cannot go over 10 kgf.

The weight and size of the Ball-Bot cannot go beyond the constraints. After consideration, the diameter of the spherical wheel is selected to be 20 cm. The size and weight of the robot body is then expected to be limited within the constraints, if the material and structure are well selected and designed.

2.1.2. Material & Manufacture

To design the structural parts of Q-Baller with right material and an economically feasible way to manufacture requires good planning and consideration in this challenging mechanical design topic. At first we planned to adopt 3D printed plastic structural parts. The idea was later abandoned since 3D printing may result in uncontrollable manufacture inaccuracy. It may also take a lot of material and time for a 3D printer to produce all the structural parts. Eventually we decide to design the structural parts that can be manufactured through sheet metal forming [7].

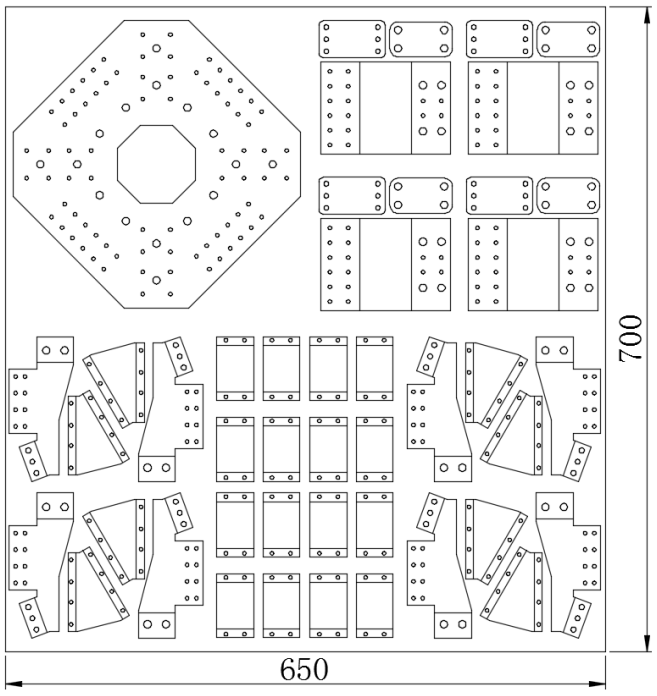


Figure 2.3: Structural Parts Allocated on a 650*700*3 mm3 Metal Sheet

Through laser cutting, drilling, pressing and bending, sheet metal parts can be manufactured through relatively simple procedures and obtaining satisfactory accuracy. The labor and material cost of parts manufactured through sheet metal forming are relatively lower than those from forging, molding and machining. The challenge of this approach is that the parts must be designed for sheet metal forming and provide sufficient strength after assembling. The final design of the structural parts, as shown in Fig. 2.3, has uniform thickness and can be compacted enclosed in a piece of metal sheet.

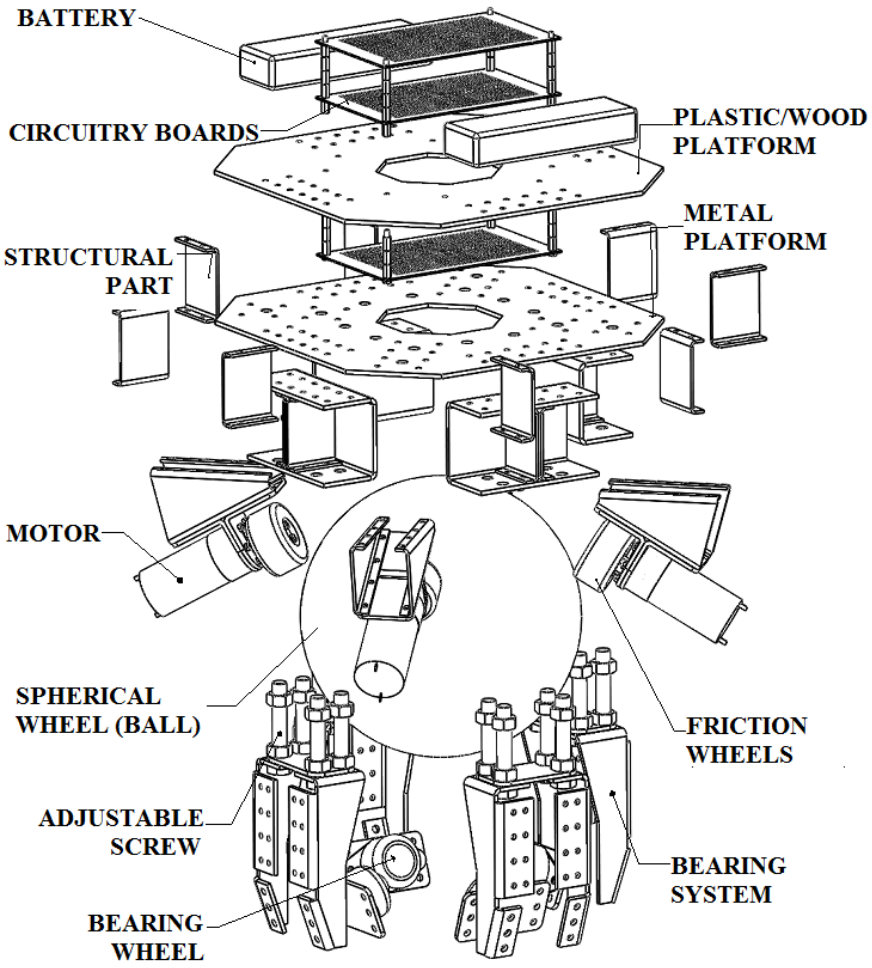


Figure 2.4: Q-Baller Spherical Wheeled Robot Design Overview

The material of the structural parts was eventually selected as Aluminum Alloy 5052 [8], which is very popular in the market and compatible for sheet metal forming. Compared with steels, aluminum alloys are lighter in density. The structure of the Ball-Bot was analyzed with Finite Element Method and the safety factor of the system is above 3.

2.1.3. Mechanical Design Overview

The mechanical design of Q-Baller is overviewed in Fig. 2.4. The robot is about $300\times300\times400$ mm^3 in size, and 8 kgf in weight. The four friction drive systems will provide symmetric control input to the Ball-Bot system. The spherical wheel was made from an steel spherical shell with a high traction coating, which would weigh about 1.5 kgf.

The four bearing systems mentioned before are attached and fastened to the robot body through the adjustable screws. The bearing wheels are made from steel spherical balls enclosed in the casing shells that can passively accompany surface translation at any direction. Thus, the bearing system can offer extra contact points to the system while also providing slots for add-ons.

2.2. Mechatronic System Design of Q-Baller

The mechatronic system of the Q-Baller consists of the control processor, the sensor system and four motors. The design aims to reach the satisfactory results with the most cost-effective mechatronic components.

2.2.1. DC Motor & Its Dynamics

DC Motor, AC Motor and Stepper Motor are the three types of motors commonly used for autonomous & robotic vehicles [9]. For our Q-Baller, we selected Permanent

Magnet DC motor due to its relatively simple and linear dynamic behavior and high efficiency when it comes to power output.

The two commonly used types of Permanent Magnet DC Motors are the Brushed DC (BDC) Motors and the Brushless DC (BLDC) Motors. The two types of motors have different structures, but the dynamic system of the motors share features in common.

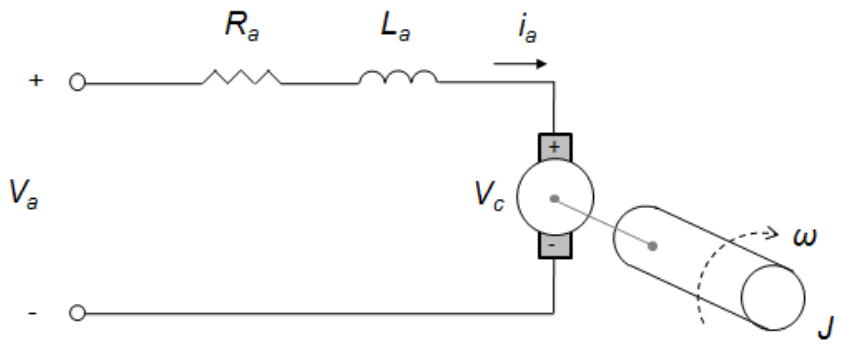


Figure 2.5: Dynamic Model of BDC Motor at Free Spin Mode

The dynamic mode of a BDC Motor's mechatronic system at Free Spin Mode is depicted in Fig. 2.5. The dynamic model of the system is derived as:

$$\begin{cases} V_a = R_a i_a + L_a \dfrac{di_a}{dt} + V_c \\ T_a = J\dot{\omega} + b\omega \end{cases} \text{, where } \begin{cases} V_c = K_e \phi_M n \\ T_a = K_t \phi_M i_a \end{cases} \tag{2.1}$$

Here V_a and i_a are the input voltage and the current of the electronic system respectively; L_a is the self-inductance of motor armature; R_a is the resistance of motor armature; V_c is the motor's Back-EMF (Back Electronic Magnetic Force) Voltage; J and b are the angular inertia and angular viscosity damper of the motor shaft respectively; and ω is the angular velocity of the shaft.

The Back-EMF Voltage V_c is generated by the inverse torque of the system output. Here in (2.1) K_e is described as the Back-EMF constant and K_t as the Torque Constant.

These constants are determined by the properties of the motor. In these equations, ϕ_M is the magnetic flux that goes through the armature, and n is the rotation rate (rounds per minute) of the shaft.

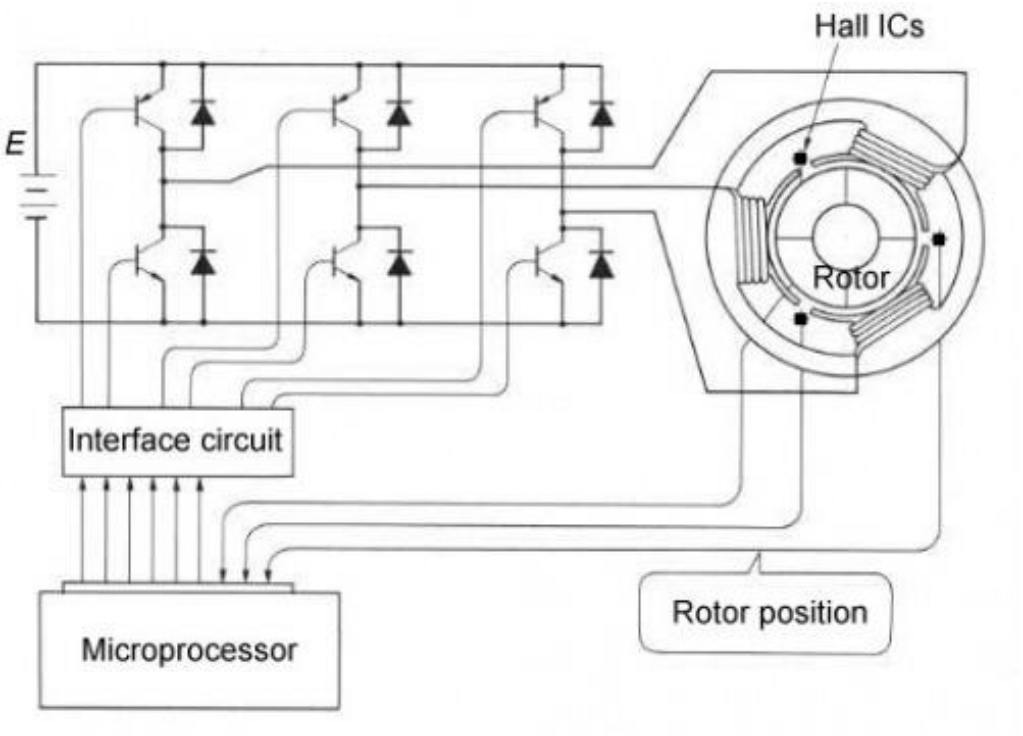


Figure 2.6: Circuitry of BLDC Motor

As shown in Fig 2.5, BLDC Motors have more complicated electronic structures [10]. These motors are usually integrated with built in controller boards, and the armatures of the motor on different phases are activated based on the regulation form the controller based on the rotor position feedback from the Hall ICs sensors.

The dynamic equations of BLDC Motor are described below:

$$\begin{cases} V_a = R_{a_X} i_{a_X} + L_{a_X} \dfrac{di_{a_X}}{dt} + V_{c_X} \\ T_a = \sum_{X=1}^{N} T_{a_X} \end{cases};$$

$$\text{where } \begin{cases} V_{c_X} = K_{e_X}\phi\left(\theta + \dfrac{2X\pi}{N}\right)n \\ T_{a_X} = K_{t_X}\phi\left(\theta + \dfrac{2X\pi}{N}\right)i_{a_X} \end{cases} \text{ and } X = 1, 2, \dots, N \tag{2.2}$$

Here N is the number of the phases of the motor; θ is the current phase angle of the motor. According to the design, the properties of the motor will also satisfy:

$$K_{e_1} = K_{e_2} = \cdots = K_{e_N} = K_e$$

$$K_{t_1} = K_{t_2} = \cdots = \mathrm{K}_{t_N} = K_t$$

$$R_{a_1} = R_{a_2} = \cdots = R_{a_N} = R_a$$

$$L_{a_1} = L_{a_2} = \cdots = L_{a_N} = L_a$$

While $\phi\left(\theta + \frac{2X\pi}{N}\right)$ is not a constant, we can roughly assume that:

$$\sum_{X=1}^{N} \phi\left(\theta + \frac{2X\pi}{N}\right) \cong \phi_M \tag{2.3}$$

The information of K_e, K_t, R_a and L_a are not provided in the product manual and can only be estimated through experiments [11]. The common properties offered in the motor manual usually includes the free spin speed (when load is zero), the stalling torque (when velocity is zero) and the standard output power.

According to the dynamic equations of the both BDC and BLDC motors, dia/dt take place as a transient effect when there is a change of input power or output load. When ignoring the dynamics of the motor (the time of transition to steady state), the steady state relationship between the rotary velocity of the motor and the torque can be simplified as:

$$n = \frac{V_a}{K_e\phi_M} - \frac{R_a}{K_e K_t \phi_M^2} T \tag{2.4}$$

Equation (2.4) shows that the steady state relationship between n and T under a constant input V_a is linear. During no load condition, we can assume that $\frac{R_a}{K_e K_t \phi_M^2} T_a = 0$. The motor will reach its free spin speed n_0:

$$n_0 = \frac{V_a}{K_e \phi_M}$$

Similarly, the motor reaches its stalling torque T_s when $n = 0$:

$$T_s = \frac{V_a K_t \phi_M}{R_a}$$

Therefore, according to the provided motor product specifications. We can simplify equation (2.4) to:

$$\frac{2\pi n_{0_0}}{60 T_{s_0}} T = \frac{2\pi n_{0_0} V_a}{60 V_{a_0}} - \omega \tag{2.5}$$

Here T_{s_0} and n_{0_0} are the stalling torque and free spinning speed at the rated voltage V_{a_0}.

By using the approximating method introduced above we can achieve equation (2.5) the characteristic equation of the motors easily without experimenting on the components. The most significant deficiency of this equation is that it does not include the transient effect of L_a. But when the current is changing slowly, the effect of L_a will not be much a concern.

When substituting V_a with voltage input symbol U, by adding the motor load into equation (2.5). we can achieve the linear relationship between motor input voltage U and output torque T_m as shown below:

$$\frac{T_{s_0} U}{U_0} - T_m \ = J\dot{\omega} + b\omega + \frac{60 T_{s_0}}{2\pi n_{0_0}} \omega \tag{2.6}$$

2.2.2. Electronic Modules

The selection of processor of the Q-Baller determines the performance of the system. The processor not only has to process the control algorithms but also to provide output signals and read in feedback data with its output peripheral.

With the development of electronic processor technology in the past several years, the controller requirement of our Q-Baller can now be satisfied by conventional embedded microchip processors. We have selected STM32F407VET6 [12], a 32-bit 168 MHz embedded processor, as the microcontroller of the Q-Baller. Compared with the controller built on the previously introduced Ball-Bot designs which are made a few years ago, the controller for Q-Baller may not be as powerful in calculation speed, but it will be far more than capable of performing the basic Ball-Bot functions – providing the control signals for the driver circuitries of the four motors, reading in sensor information and communicating with exterior devices through wireless transmitters.

In order to fully observe the dynamics of Q-Baller, the sensor system of Ball-Bot includes:

1) An integrated sensor of gyroscopic sensor and accelerometer which can feed back the acceleration, angular velocity and the attitude of the system.
2) Four encoders that reads the angular position of the motors.

To establish communication between Q-Baller and other systems, such as computers, wireless transmitters, are adopted regarding the communication distance and speed requirement. Bluetooth devices are easy to use and can realize data transmission within a close range. When the robot are to be control from a terminal far away, more powerful wireless transmitters are considered better options.

As mentioned in the previous chapter, if BLDC motors are adopted in the friction drive systems of Q-Baller, motor controllers are usually integrated within the motor so they do not require any exterior motor drivers. For BDC motors, motor driver modules are required to regulate the powers and directions of the motors.

In addition to the modules mentioned above, there are other components in the electronic system that is crucial to the operation of the system, such as the voltage regulators and power regulators which will provide appropriate power supply to different modules.

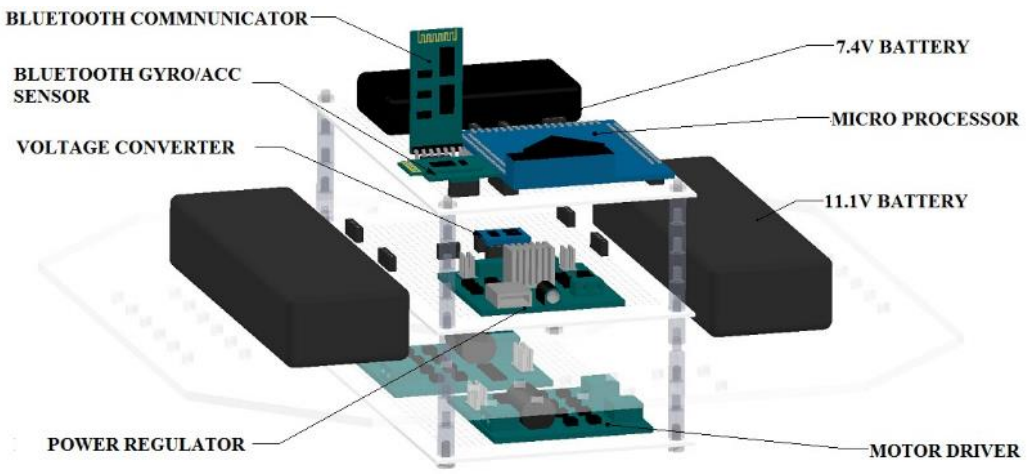


Figure 2.7: Preliminary Electronic System

Finally, battery packs of the system are chosen based on the power requirements of the controller and system output. The overview of the preliminary electronic system design is presented in Fig. 2.7.

2.3. Conclusion

In this chapter, a Ball-Bot designed named “Q-Baller” is introduced. We discussed the design ideas and techniques that lead to the detailed mechanical design of Q-Baller.

The dynamic characteristics of the adopted motors are analyzed in detail in preparation for the dynamic modeling and control of the Q-baller system in the following chapters. Finally, the selections of electronic components will pave the road to the embedded system development introduced in later chapters.

CHAPTER 3. DYNAMIC MODELING & ANALYSIS

The mechatronic design of Q-Baller has defined the system properties for us to establish its dynamic model. The modeling is based on the combined characteristics of both mechanical and electronic properties. Models in different forms will be established for different control purposes. The model of Q-Baller will go through zero-input stability and controllability analysis. The dynamic analysis of Q-baller will be crucial for the design and implement of suitable controller.

3.1. Coordinate System and Vector Definition

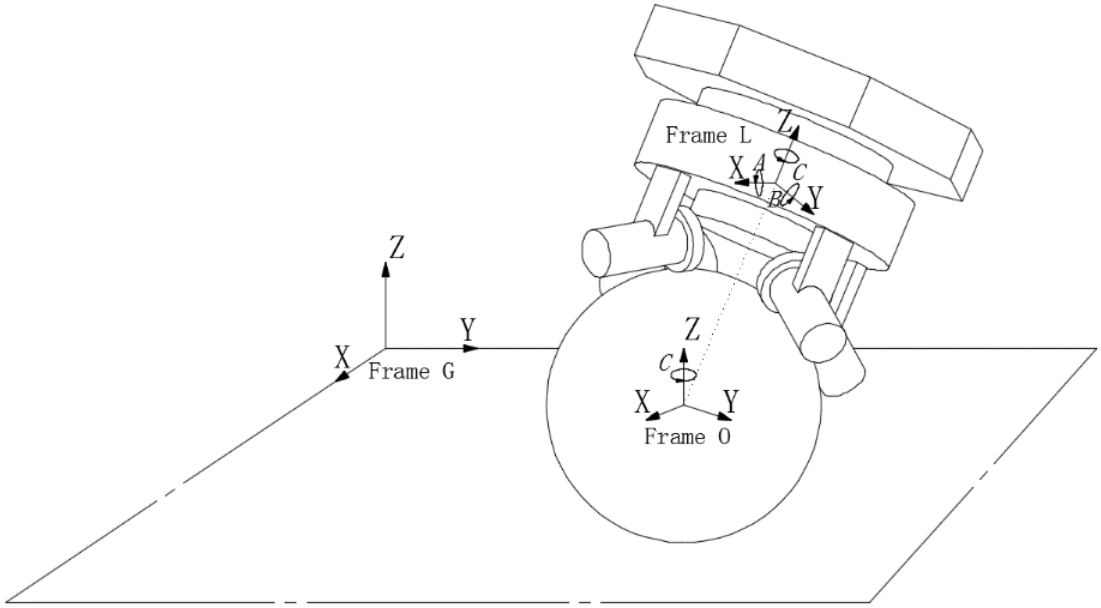


Figure 3.1: Different Coordinate Frames of the System

Q-Baller contains 2 rigid body parts. However, the dynamics of Q-baller is not simple because of the 4 friction drive we used. The motion of the spherical wheel is

separated from the robot body, indicating that different coordinate systems should be used for the analysis of motion.

The spherical wheel is considered perfectly symmetrical to any planes or lines that goes through its center. Since the robot body system is constantly riding on the spherical wheel when it is working without malfunctions, the origin of the robot body has been moved to the center of the ball from the its center of mass.

After careful considerations, the Cartesian coordinate frames of the system used in the modeling process are depicted in Fig. 3.1. All of the three frequently used coordinate systems are selected according to attitude of the robot body, which are introduced with detail below [13]:

1) The Ground Frame (Frame G) will be set up by the time the robot starts running, which is the absolute coordinate system. The origin of the Frame G is set at an arbitrary point in the ground near Q-Baller.

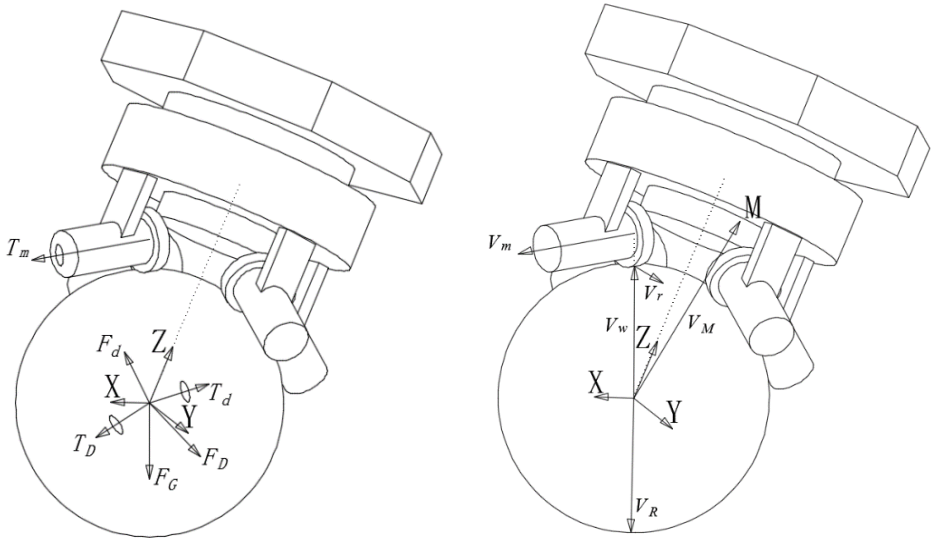


Figure 3.2: Force and Geometric Vectors of the System

2) The origin of the Orientation Frame (Frame O) is constantly settled at the center of the spherical wheel. Frame O takes form after the attitude transition of the Q-Baller body around the Z Axis (Yawing) from Frame G.
3) The Local Frame (Frame L) takes form after the Pitching (Y Axis) and Rolling (X Axis) of Q-Baller body from Frame O. Its center is also located at the center of the ball. In Fig.3.1 Frame L is moved away from the origin for clearer presentation purpose.

The Q-baller system includes multiple geometric and input parameters which will be used during the modeling process. The vectors used in the system modeling as presented in Fig. 3.2 are defined below:

1) $\overrightarrow{V_M}$ stands for the vector pointing from the origin of Frame L to the center of mass of the robot body:

$$\overrightarrow{V_M} = [H_X \quad H_Y \quad H_Z]^T$$

2) $\overrightarrow{V_R}$ and $\overrightarrow{V_w}$ points from the origin to the contact point of the ground and to each of the Omni-Wheels in the system respectively:

$$\overrightarrow{V_R} = [0 \quad 0 \quad -R]^T;\ \overrightarrow{V_w} = r * \left\{ \begin{bmatrix} ib \\ -ib \\ ic \end{bmatrix} \ \begin{bmatrix} ib \\ ib \\ ic \end{bmatrix} \ \begin{bmatrix} -ib \\ ib \\ ic \end{bmatrix} \ \begin{bmatrix} -ib \\ -ib \\ ic \end{bmatrix} \right\}$$

3) $\overrightarrow{V_m}$ and $\overrightarrow{V_r}$ stand for the positive direction of the rotary velocity and the positive translational velocity direction at the contact point on the Omni-Wheel from one of the motors respectively:

$$\overrightarrow{V_m} = \left\{ \begin{bmatrix} ix \\ -iy \\ -iz \end{bmatrix} \ \begin{bmatrix} -ix \\ -iy \\ iz \end{bmatrix} \ \begin{bmatrix} -ix \\ iy \\ -iz \end{bmatrix} \ \begin{bmatrix} ix \\ iy \\ iz \end{bmatrix} \right\};\ \overrightarrow{V_r} = \left\{ \begin{bmatrix} ia \\ ia \\ 0 \end{bmatrix} \ \begin{bmatrix} ia \\ -ia \\ 0 \end{bmatrix} \ \begin{bmatrix} -ia \\ -ia \\ 0 \end{bmatrix} \ \begin{bmatrix} -ia \\ ia \\ 0 \end{bmatrix} \right\}$$

4) $\overrightarrow{\mathrm{T_m}}$ is the actuating torque effect on the robot body from one of the motors:

$$\overrightarrow{T_m} = \left\{ \begin{bmatrix} ix \\ -iy \\ -iz \end{bmatrix} \quad \begin{bmatrix} -ix \\ -iy \\ iz \end{bmatrix} \quad \begin{bmatrix} -ix \\ iy \\ -iz \end{bmatrix} \quad \begin{bmatrix} ix \\ iy \\ iz \end{bmatrix} \right\}$$

5) $\overrightarrow{\mathrm{F_G}}$ stands for the gravity force of the Ball-Bot's body (origins from the Center of Mass of the body):

$$\overrightarrow{F_G} = [0 \quad 0 \quad -g * M]^T$$

6) $\overrightarrow{\mathrm{T_D}}, \overrightarrow{\mathrm{F_D}}, \overrightarrow{\mathrm{T_d}}, \overrightarrow{\mathrm{F_d}}$ are the 4 exterior perturbation inputs (forces and torques) acting on the body and the ball (which are under Frame G) which will be used as process noises during simulation:

$$\overrightarrow{T_d} = \begin{bmatrix} T_{dx} \\ T_{dy} \\ T_{dz} \end{bmatrix}; \quad \overrightarrow{F_d} = \begin{bmatrix} F_{dx} \\ F_{dy} \\ F_{dz} \end{bmatrix}; \quad \overrightarrow{T_D} = \begin{bmatrix} T_{Dx} \\ T_{Dy} \\ T_{Dz} \end{bmatrix}; \quad \overrightarrow{F_D} = \begin{bmatrix} F_{Dx} \\ F_{Dy} \\ F_{Dz} \end{bmatrix}$$

In the definitions of the vectors above, R is the radius of the ball and r is the radius of the ball, which are scalar values; M is the mass of the robot body; m is the mass of the spherical wheel; ix, iy, iz, ia, ib and ic are intermediate geometric scalar constants (single values). Since there are four friction drive systems, each column of the $\overrightarrow{\mathrm{V_w}}$, $\overrightarrow{\mathrm{V_m}}$, $\overrightarrow{\mathrm{V_r}}$ and $\overrightarrow{\mathrm{T_m}}$ presents the information for one of the four friction drive system respectively.

3.2. Dynamic Modeling

The Q-Baller motion is defined in the 3D space due to the coupling effect of different attitude states and the application purpose of the robot. For this reason, Q-Baller's dynamic system is modeled through Lagrangian Method. Since modeling is crucial for dynamic

analysis and controller design, we established a flexible and detailed model by taking exclusive information from the system's mechatronic model into consideration.

To adopt a Lagrangian Method [14], we first choose the generalized coordinates according to the following assumptions:

1) The Ball-Bot will always be on the ground;
2) There is no slipping between any of the contact surfaces;
3) The body never loses contact with the ball;

Q-Baller is a two-body system. However, all of the states of the two objects can be expressed by the coordinate vector which only includes 5 states in total:

$$q = [A \quad B \quad C \quad X \quad Y]^T \tag{3.1}$$

As shown in (3.1), A, B, C are the Euler Angles describing the rotation of the robot body around axis X (Rolling), Y (Pitching), Z (Yawing) in Frame L, and X, Y are the accumulated distance of the Ball-Bot traveled along X and Y directions in Frame O which are derived from the rotation of the ball. The position and velocity vectors be represented with these five states. Here are some exemplary intermediate terms defined for the Lagrange Equations:

1) Angular velocity of the robot body (symbolized with B) in Frame L:

$$W_{B-L} = M_J[\dot{A} \quad \dot{B} \quad \dot{C}]^T \tag{3.2}$$

2) Angular velocity of the spherical wheel (symbolized with b) in Frame O:

$$W_{b-O} = \left[-\frac{\dot{Y}}{R} \quad \frac{\dot{X}}{R} \quad 0\right]^T \tag{3.3}$$

3) Translational velocity of the whole robot system in Frame G converted from the translational velocity in Frame O:

$$V_G = C_O^G V_O = C_O^G * [\dot{X} \quad \dot{Y} \quad 0]^T \tag{3.4}$$

In the vector definitions presented above, C_x^y stands for the coordinate transformation matrix from Frame x to Frame y. For example, C_O^G is the coordinate conversion matrix from Frame O to Frame G. M_J is the Jacobian Matrix of C_G^L, which can transform the Euler Angle velocities to the true angular velocities of the robot body. Some of these conversion matrixes are shown below in (3.5), (3.6) [15][16]:

$$M_J = \begin{bmatrix} 1 & 0 & -sin(B) \\ 0 & cos(A) & sin(A)\,cos(B) \\ 0 & -sin(A) & cos(A)\,cos(B) \end{bmatrix} \tag{3.5}$$

$$C_L^G = \begin{bmatrix} cos(C) & -sin(C) & 0 \\ sin(C) & cos(C) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(B) & 0 & sin(B) \\ 0 & 1 & 0 \\ -sin(B) & 0 & cos(B) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(A) & -sin(A) \\ 0 & sin(A) & cos(A) \end{bmatrix} \tag{3.6}$$

According to the principles of Lagrangian Mechanics, we have established the following equations that govern the motion of the Ball-Bot's mechanical system [17][18] based on the system energy features:

1) The translational kinetic energy of the whole robotic system:

$$T_L = \frac{1}{2} V_G^T (M + m) V_G \tag{3.7}$$

2) The sum of rotary kinetic energies of the robot body and spherical wheel:

$$T_R = \frac{1}{2} (W_{B-L}^T J_B W_{B-L}) + \frac{1}{2} (W_{b-O}^T J_b W_{b-O}) \tag{3.8}$$

3) The coupling kinetic energy which occurs since the origin of rotation of the robot body does not locate at its center of mass:

$$T_C = M(V_G^T C_L^G)\left(W_{B-L} \times \overrightarrow{V_M}\right) \tag{3.9}$$

4) The potential energy from the non-conservative forces – the gravitation forces:

$$V = -F_G C_L^G \overrightarrow{V_M} \tag{3.10}$$

Therefore, we can get the sum of the mechanic energy:

$$L = T_L + T_R + T_C - V \tag{3.11}$$

To establish the Lagrange Equation for the dynamic system, we also need to calculate the virtual works that input into or output from the system:

$$Q_I = W_{b-O}^T C_L^O (T_W + C_G^L T_d) + W_{B-L}^T (-T_W + C_G^L T_D) + V_G^T (F_D + F_d) \tag{3.12}$$

We also considered the energy dissipation based on Rayleigh Energy Dissipation Theory:

$$Q_D = \frac{1}{2} V_G^T (C_L^G B_{BL} + B_{bL}) V_G + \frac{1}{2} (W_{B-L}^T B_{BR} W_{B-L}) + \frac{1}{2} (W_{b-O}^T B_{bR} W_{b-O}) \tag{3.13}$$

Finally, the Lagrange Equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) = \left(\frac{\partial (Q_I dt)}{\partial (dq_i)}\right) - \left(\frac{\partial Q_D}{\partial \dot{q}_i}\right) \quad (for \;\; i = 1,2 \dots 5) \tag{3.14}$$

In the equations listed above, J_x matrixes are the 3×3 angular inertia matrixes of the bodies (x as B for the robot body and x as b for the spherical wheel as mentioned before); B_{xL} stand for the 3×3 translational viscosity dampers; and B_{xR} represent the 3×3 angular viscosity dampers. J_B, B_{BL} and B_{BR} are all measured in the Frame L, and

J_b, B_{bL} and B_{bR} are diagonal matrixes with identical elements due to the symmetricity of the spherical wheel.

The following part is the modeling of the mechatronic system and the constraints to complete the modeling of the whole mechatronic system. According to the geometry of the robot system, the working torque T_W as the sum of the motor outputs acting on the ball:

$$T_W = -\frac{R}{r}\begin{bmatrix} T_{m_{x+y+}} & T_{m_{x+y-}} & T_{m_{x-y-}} & T_{m_{x-y+}} \end{bmatrix}^T \tag{3.15}$$

The subscript of the motor's torque T_m indicates the direction the torque tends to make the whole robot move to in the Frame O. Each T_m is determined by the dynamics of the DC motor as mentioned before in Chapter 2:

$$\frac{T_{s_0}U}{U_0} - T_m = J_w\dot{\omega} + b_w\omega + \frac{60T_{s_0}}{2\pi n_{0_0}}\omega \tag{3.16}$$

Equation (3.16) has omitted the fast dynamics from the inductances in the motor, and it is simplified by the free spin velocity n_{0_0}, stalling torque T_{s_0} and nominal voltage U_0,according to the typical steady performance plot of DC motors.

Based on the nonslip condition, the contact point velocity at the friction wheel and the ball should be identical. Thus, for each motor the velocity equation should be:

$$\omega_j = V_{r_j}^T((C_O^L W_{b-O})\times V_{W_j} - W_{B-L}\times V_{W_j})/r \tag{3.17}$$

Where V_{XX_j} is the jth column of the V_{XX} matrix, and $j = 1,2,3,4$. The equations couple the electronic systems of the motors with the mechanical system of the robot.

3.3. Standard Presentation of Dynamic System

Previous system modeling layout indicated that the modeling process can cater to the special need of analyzing system with modeling inaccuracy, since we have allowed deviation of the center of mass, asymmetric rotary inertia and other flexibilities.

Due to the complexity of the system, the modeling process was realized through computer calculation tools. The calculated governing equation is too long be attached in the thesis, but the modeling algorithm will be included in the appendix.

While the for the complete system, we can assume that the states and the inputs are those as shown below:

$$x = \begin{bmatrix} A & B & C & X & Y & \dot{A} & \dot{B} & \dot{C} & \dot{X} & \dot{Y} \end{bmatrix}^T \tag{3.18}$$

$$u = [U_{x+y+} \quad U_{x+y-} \quad U_{x-y-} \quad U_{x-y+}]^T \tag{3.19}$$

$$v = [T_D^T \quad F_D^T \quad T_d^T \quad F_d^T]^T \tag{3.20}$$

The standard presentation of the system is:

$$\dot{x} = f(x, u, v) \tag{3.21}$$

$$y = g(x, u, w) \tag{3.22}$$

Here, $f(x, u, v)$ is the set of the ODE functions that calculates the derivatives of the true state x, where v is the process noise (or exterior disturbances) that will affect the dynamic system; $g(x, u, w)$ is the set of the observer functions to calculate the observed states y, where w is the observation noise that would only affect the feedbacks of the observer.

The standard system functions describe the system as a whole – all dynamic information (structure, size, driver, etc.) are included in the ODE function, and the observer functions contains the information of the sensors. The nonlinear system is highly coupled and can hardly be analyzed directly through the equations of motions. To solve this problem, State-Space Presentation is adopted through linearizing the system to the following format [19]:

$$\dot{X_{SS}} = A_{SS}X_{SS} + B_{SS}U_{SS} + E_{SS}V_{SS} \tag{3.23}$$

$$Y_{SS} = C_{SS}X_{SS} + D_{SS}U_{SS} + F_{SS}W_{SS} \tag{3.24}$$

Here V_{SS} and W_{SS} are process noise and measurement noise, respectively. The State-Space equations above include process and measurement noises. For a model that operates in an idealistic environment (without noises), the State-Space equations are simplified as below:

$$\dot{X_{SS}} = A_{SS}X_{SS} + B_{SS}U_{SS} \tag{3.25}$$

$$Y_{SS} = C_{SS}X_{SS} + D_{SS}U_{SS} \tag{3.26}$$

In the State-Space Matrix, C_{SS} is determined through control objective. For example, the corresponding Y_{SS} in the controllers for position control and velocity control are shown below:

$$Y_{Position} = \begin{bmatrix} A & B & C & X & Y & \dot{A} & \dot{B} & \dot{C} & \dot{X} & \dot{Y} \end{bmatrix}^T$$

$$Y_{Velocity} = \begin{bmatrix} A & B & C & \dot{A} & \dot{B} & \dot{C} & \dot{X} & \dot{Y} \end{bmatrix}^T$$

3.4. Characteristic of the Standard Model

The modeling process introduced before allow generations of varies models for different experiment and simulation purpose. Therefore, a standard model is selected to represent the general characteristics of the Q-Baller. The Standard model has symmetric physical properties and most of its features are idealistic. The system parameters we selected for the standard model are shown below in Figure 3.3, which are selected according to the 3D model designed in the CAD software.

Height of the Center of Mass (H)	0.107 m	Mass of the Ball (m)	1.5 kg
Rotary Inertia of Omni-Wheel (J_W)	$1.25*10^{-5}$ kg-m^2	Radius of the Ball (R)	0.1 m
Mass of the Body (M)	6.4 kg	Radius of the Omni-Wheels (r)	0.024m
Damper on Body during Translation (b_{BL})	0.008 N-s/m	Damper on Ball during Translation (b_{bL})	0.002 N-s/m
Rotary Inertia of Ball (J_b)	0.00975 kg-m^2	Rotary Damper on Ball (b_{bR})	0.01 N-s/rad
Rotary Inertia of Body on X axis (j_{XX})	0.1488 kg-m^2	Rotary Damper on Body on X axis (b_{XX})	0.02 N-s/rad
Rotary Inertia of Body on Y axis (j_{YY})	0.1512 kg-m^2	Rotary Damper on Body on Y axis (b_{YY})	0.02 N-s/rad
Rotary Inertia of Body on Z axis (j_{ZZ})	0.0746 kg-m^2	Rotary Damper on Body on Z axis (b_{ZZ})	0.01 N-s/rad
Standard Stalling Torque of Motor (T_{S0})	2.5 N-m	Standard No Load Speed of Motor (n_{00})	2000 r/min
Standard Voltage of Motor (U_0)	12V	Rotary Damper on Omni-Wheel (b_W)	0.005 N-s/rad

Figure 3.3: Properties of the Standard Model

As the system has multiple equilibrium points and the states are highly coupled, single linearization will not be able to show the overall feature of the system. Through observation at several of its linearization points, it is easy to realize the high nonlinearity of the system due to the coupled states. For example, when linearized at an arbitrary point (not equilibrium):

$$x = [0.0873 \quad 0.0873 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

$$u = [0 \quad 0 \quad 0 \quad 0]^T$$

The State-Space matrixes A_{SS} and B_{SS} reflect the mentioned features clearly, as can be recognized from the data presented below:

$$A_{SS} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 68.15 & -0.04 & 0 & 0 & 0 & -8.17 & 0.02 & 0.16 & -0.20 & -80.65 \\ -1.05 & 66.02 & 0 & 0 & 0 & 0.03 & -7.96 & 0.14 & 78.42 & 0.20 \\ 5.80 & 5.84 & 0 & 0 & 0 & -0.005 & -0.34 & -3.99 & 3.43 & -3.43 \\ 0.12 & -4.97 & 0 & 0 & 0 & -0.007 & 1.10 & 0.05 & -11.02 & -0.03 \\ 5.12 & -0.08 & 0 & 0 & 0 & -1.12 & 0.003 & 0.05 & -0.03 & -11.18 \end{bmatrix}$$

$$B_{SS} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 8.76 & -8.58 & -6.45 & 1.25 & 1.26 \\ 0 & 0 & 0 & 0 & 0 & -8.83 & -10.10 & 4.83 & 1.37 & -1.26 \\ 0 & 0 & 0 & 0 & 0 & -10.38 & 10.17 & -4.85 & -1.38 & -1.39 \\ 0 & 0 & 0 & 0 & 0 & 10.44 & 8.50 & 6.47 & -1.23 & 1.39 \end{bmatrix}^T$$

However, at the point where all of the states are zero, the State-Space matrixes A_{SS} and B_{SS} has shown weak coupling effect and nonlinearity, which are shown below:

$$A_{SS} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 69.27 & 0 & 0 & 0 & 0 & -8.23 & 0 & 0 & 0 & -81.08 \\ 0 & 67.60 & 0 & 0 & 0 & 0 & -8.04 & 0 & 79.12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3.98 & 0 & 0 \\ 0 & -5.17 & 0 & 0 & 0 & 0 & 1.11 & 0 & -11.12 & 0 \\ 5.30 & 0 & 0 & 0 & 0 & -1.13 & 0 & 0 & 0 & -11.27 \end{bmatrix}$$

$$B_{SS} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 9.68 & -9.44 & -5.68 & 1.32 & 1.34 \\ 0 & 0 & 0 & 0 & 0 & -9.68 & -9.44 & 5.68 & 1.32 & -1.34 \\ 0 & 0 & 0 & 0 & 0 & -9.68 & 9.44 & -5.68 & -1.32 & -1.34 \\ 0 & 0 & 0 & 0 & 0 & 9.68 & 9.44 & 5.68 & -1.32 & 1.34 \end{bmatrix}^T$$

This result indicates that the motion of Q-Baller can be handled more easily when it is close to the point where all states are zero – the zero point. Linear Controller will have better performance when closed to the zero point, while it may lose robustness when the states are away from the zero point.

3.5. Controllability and Observability

The controllability and observability analysis [19][20] are useful to test if the model, or in other words, the design satisfies the control prerequisites. The controllability of Q-Baller may vary along with the change of states. From direct observation, we can easily realize that if the robot's Pitch and Roll exceed certain extend, the friction drives will not be able to recover the system to stabilization. Therefore, we presume the system to controlled closed to the zero point where the system can be more easily stabled.

According to the Controllability Theory, the rank of the controllability matrix of the Q-Baller is presented below:

$$rank(Q_c) = rank([B_{SS} \quad A_{SS}B_{SS} \quad A_{SS}^2B_{SS} \quad \cdots \quad A_{SS}^{n-1}B_{SS}]) = 10 \qquad (3.27)$$

Here, $n = 10$, and the result yielded from (3.27) is also 10. Therefore:

$$rank(Q_c) - n = 0 \qquad (3.28)$$

This indicates that the system is fully controllable, leading to the acknowledged feasibility of the design model.

The observability of the model is judged from a more practical way, which is related to the system's actual setup: The electronic system of Q-Baller will support at least a Gyroscopic Sensor and an Accelerometer, which will detect $Y_{Sensor} = [A \quad B \quad C \quad \dot{A} \quad \dot{B} \quad \dot{C} \quad \ddot{X} \quad \ddot{Y}]^T$. Then we can achieve the $Y_{Position}$ through numerical integration. Simply from this view point we can draw the conclusion that the system is fully observable.

However, the sensor feedbacks may not be ideally reliable. For example, the state $X, Y, \dot{X}$ and $\dot{Y}$ achieve through Accelerometer may not be fully reliable due to the calculation error during integration and the signal noise. In a more realistic way, the actual observability of the system is related to the reliability of the feedback.

3.6. Conclusion

Chapter 3 has introduced the dynamic modeling of the mechatronic system of Q-Baller. The modeling is thorough including almost all features (Motors, Friction Drives, Viscosities etc.) of Q-Baller. The final model is highly complex and nonlinear, which requires us to get familiar with it through experimenting rather than only looking at the governing equations. The system shows low nonlinearity when the states are all zero (the zero point). Some simple analysis about the controllability and observability of the system are performed to prove that the system is fully controllable around the zero point.

CHAPTER 4. LINEAR CONTROL SYSTEM STUDY & DESIGN

The Dynamics of Q-Baller has been studied in the previous chapter, indicating the feasibility of the current design. Control analysis and controller design relies on the previous results and they will also elevate the dynamics study of Q-Baller to a higher level. We will discuss the stability analysis of Q-Baller through its linearized form and compare the performances of several different controllers designed through linear controller design tools for position and velocity control.

4.1. Fundamentals of Stability Analysis

Achieved from previous dynamic analysis, Q-Baller's system model can be described as below according to equation (3.21) and (3.22) (noise are excluded):

$$x = \begin{bmatrix} A & B & C & X & Y & \dot{A} & \dot{B} & \dot{C} & \dot{X} & \dot{Y} \end{bmatrix}^T \tag{4.1}$$

$$u = [U_{x+y+} \quad U_{x+y-} \quad U_{x-y-} \quad U_{x-y+}]^T \tag{4.2}$$

$$v = [T_D^T \quad F_D^T \quad T_d^T \quad F_d^T]^T \tag{4.3}$$

$$\dot{x} = f(x, u) \tag{4.4}$$

$$y = g(x, u) \tag{4.5}$$

From Lyapunov Stability Criterion [21], if Q-Baller is stable in a certain domain according $x \in D$ to a certain equilibrium point, for an energy function:

$$V = x^{\mathrm{T}} P x \tag{4.6}$$

The system satisfies the following conditions:

1) P is positive definite;

2) In domain D, we have:

$$\dot{V} = x^T P\dot{x} + \dot{x}^T Px + x^T \dot{P}x = x^T Pf(x,u) + f(x,u)^T Px + x^T \dot{P}x \leq 0 \quad (4.7)$$

Due to the complexity of Q-Baller System, we have to replace $f(x,u)$ with the system's State-Space Equations according to (3.25) and (3.26):

$$\dot{X_{SS}} = A_{SS}X_{SS} + B_{SS}U_{SS} \quad (4.8)$$

$$Y_{SS} = C_{SS}X_{SS} + D_{SS}U_{SS} \quad (4.9)$$

Therefore, with the following additional conditions:

1) P is positive definite constant matrix;

2) High order states and external noise are eliminated, and $D_{SS} = 0$;

3) A linear controller is designed for the system:

$$U = -KC_{SS}x \quad (4.10)$$

Function (4.7) can be further simplified to the following state:

$$\dot{V} = x^T(P(A_{SS} - B_{SS}KC_{SS}) + (A_{SS} - B_{SS}KC_{SS})^T P)x \quad (4.11)$$

$$Q = -(P(A_{SS} - B_{SS}KC_{SS}) + (A_{SS} - B_{SS}KC_{SS})^T P) \quad (4.12)$$

From (4.11) and (4.12) it is obvious that to fulfill the requirement of system stability, we must have Q as a symmetric positive definite matrix in domain $x \in D$.

For the convenience purpose, we define the Lyapunov Function to be:

$$\mathcal{L}(A_{\mathcal{L}}, x, Q_{\mathcal{L}}) = P_{\mathcal{L}} \quad (4.13)$$

Here, $Q_{\mathcal{L}}$ is the symmetric positive definite matrix, and $x \in D$ indicating that the Function only works on domain D.

4.2. Linear Controller Design at Zero Point

As discussed before, the highly nonlinear and state-coupling characteristic if Q-Baller cannot be easily controlled with a linear controller designed at the zero point. The Gain-Scheduling method provide solution by applying different controller to the system at different states.

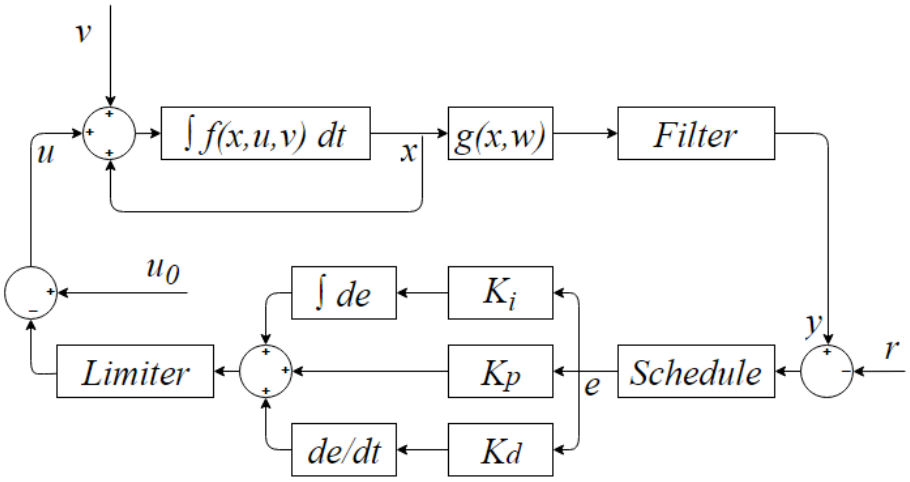


Figure 4.1: Control System Flowchart with Linear Controller

The application of a linear controller to our design is depicted in Fig. 4.1. For each iteration, the observed states will be filtered and compared with the objective reference [22]. The observed the states will also determine the scheduled controller for input update in this iteration. The error will then be transferred to the controller, updating the input through applying error to the Proportional, Integral and Differential (PID) controller respectively [19][23]:

$$U_P = K_P e \tag{4.14}$$

$$U_I = K_I \int e(t)dt \tag{4.15}$$

$$U_D = K_D \frac{\partial e(t)}{\partial t} \tag{4.16}$$

In addition to the P Controller which updates the input according to the current state, I Controller tends to eliminate steady state error, and D Controller improves the stability with the tendency of the states.

The Limiter in the Figure 4.1. will limit the input to practical value according the system properties such as input boundaries and maximum input variation limitations.

To adopt the Gain-Scheduling Method, it requires us to design controller at multiple equilibrium states. While the other equilibrium points may be hard to acquire through analytical method due to the complexity of the system, we decide to start from the zero point, and then acquire other equilibrium points through experimental means.

Linear Quadratic Regulator (LQR) [24][25] is selected as the original linear controller designer for Q-Baller. LQR controllers are optimal controllers which minimizes the cost function that balances the energy cost and system performance:

$$J = \frac{1}{2}\int_0^\infty x(t)^T Q_{LQR} x(t) + u(t)^T R_{LQR} u(t)\ \ dt \tag{4.17}$$

Here diagonal positive definite matrixes Q_{LQR} and R_{LQR} weighs the performance and energy cost respectively. The controller of LQR is described as:

$$u(t) = -R^{-1} B_{SS}^T P_{LQR} x(t) \tag{4.18}$$

Where P_{LQR} is decided through the matrix algebraic Riccati Equation:

$$-P_{LQR} A_{SS} - A_{SS}^T P_{LQR} - Q_{LQR} + P_{LQR} B_{SS} {R_{LQR}}^{-1} B_{SS}^T P_{LQR} = 0 \tag{4.19}$$

Through computer tools like MATLAB, the controller can be easily generated. The Controller for the standard model at the zero point is shown in (4.20).

$$K_{ZP} = \begin{bmatrix} 7.381 & -7.399 & -0.500 & -0.316 & -0.316 & 1.035 & -1.046 & -0.272 & -4.333 & -4.333 \\ -7.381 & -7.399 & 0.500 & -0.316 & 0.316 & -1.035 & -1.046 & 0.272 & -4.333 & 4.333 \\ -7.381 & 7.399 & -0.500 & 0.316 & 0.316 & -1.035 & 1.046 & -0.272 & 4.333 & 4.333 \\ 7.381 & 7.399 & 0.500 & 0.316 & -0.316 & 1.035 & 1.046 & 0.272 & 4.333 & -4.333 \end{bmatrix} \quad (4.20)$$

For K_{ZP}, the LQR matrix Q and R are determined as below:

$$Q_{LQR} = diag([100 \;\; 100 \;\; 50 \;\; 20 \;\; 20 \;\; 50 \;\; 50 \;\; 25 \;\; 10 \;\; 10])$$

$$R_{LQR} = diag([50 \;\; 50 \;\; 50 \;\; 50])$$

The stability of Q-Baller is not guaranteed if LQR is not guaranteed. , Since for the controllability of the standard model at zero point we can easily test if there exists a pair of $P_{\mathcal{L}}$ and $Q_{\mathcal{L}}$ from (4.12) that satisfies the Lyapunov Stability criterion. After select $Q_{\mathcal{L}}$ as $I_{10\times10}$, the eigenvalues of the $P_{\mathcal{L}}$ that satisfies the stability criterion function $\mathcal{L}(A_{\mathcal{L}}, x, Q_{\mathcal{L}}) = P_{\mathcal{L}}$ are [21]:

$$eig(P_{\mathcal{L}}) = [0.015 \quad 0.015 \quad 0.052 \quad 0.320 \quad 0.320 \quad 1.058 \quad 2.477 \quad 2.487 \quad 9.181 \quad 9.211] > 0$$

Since all eigenvalues of P are positive - indicating that P is a positive definite, controller is proved feasible to stabilize the system in a region close to the linearization point.

4.3. Linear Controller Performance Study at Zero Point

Different control system has different characteristics. For the Q-Baller system, the controller generated in the previous subchapter is not perfect. By studying the controller performance, we can conclude to a preliminary control strategy for this system.

As obtained in the previous chapter, K_{ZP} covers 10 states in total, which are the position and velocity states of the 5 basic states. In this case, K_{ZP} is both the PD controller for the position states and the PI controller for the velocity states.

After several preliminary simulations, we have the following deductions by studying the dynamic behavior of the system:

1) A and B also provides the acceleration for X and Y, which is very useful for trajectory tracking and object following. This can be proved by analyzing the system with Newtonian Mechanics Theory.
2) PD controllers are preferable for all states during position control, while integral controllers should be implemented to state C, X and Y for velocity control. This is due to the nonlinearity of the system and factors such as viscosity dampers that the static error of velocity controls in most cases are not zero.
3) Input limitation should be applied for the system to translate with moderate behavior and avoid abruptions. The limitations include constraining the voltage changes within 12V/s and setting the cap for voltage as $\pm 12\text{V}$.

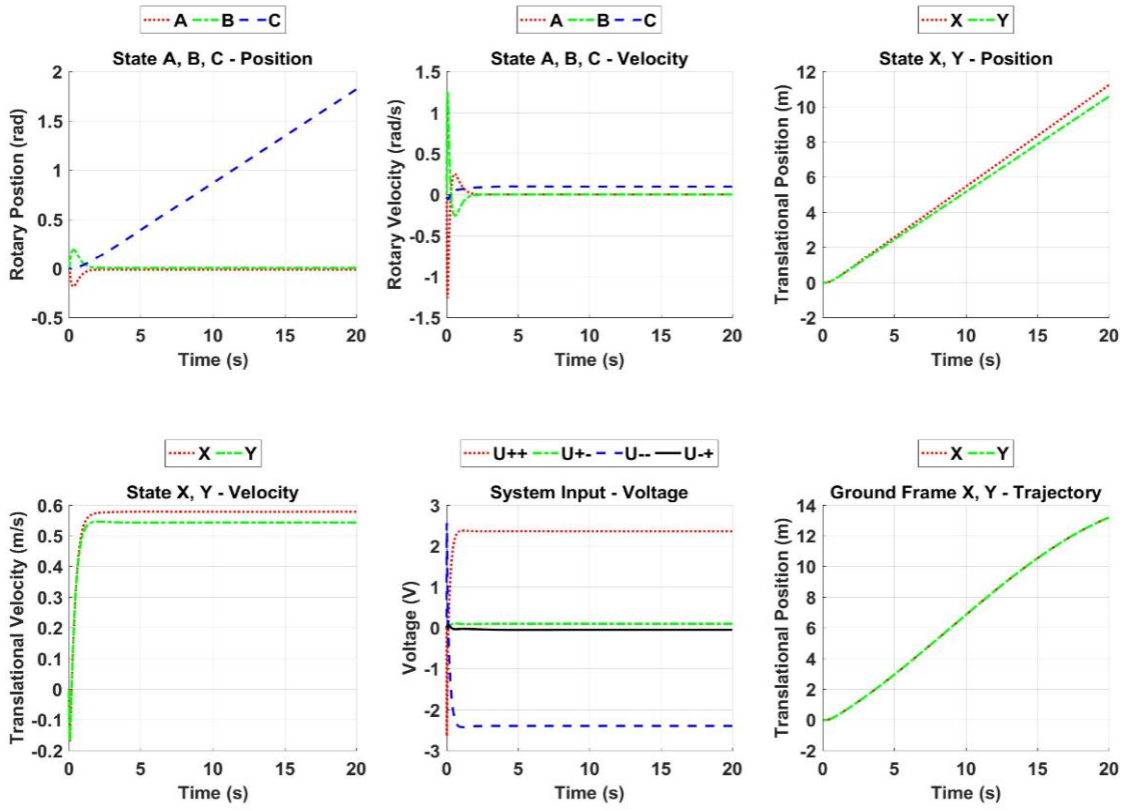
The deductions will be supported through experiments in the following chapters.

4.3.1. Implementation of PI Controller for Velocity Control

Without I controller, giving that state A and B are still under position control, the controller for the states is presented as below:

$$u = K_{PD\,4\times4}[e_A \quad e_B \quad e_{\dot{A}} \quad e_{\dot{B}}]^T + K_{P\,4\times3}[e_{\dot{C}} \quad e_{\dot{X}} \quad e_{\dot{Y}}]^T + u_r \qquad (4.21)$$

Experiment 4.1: P Velocity Controller Simulation based on LQR Controller	
Tracking Reference	Starting from the zero point; Track $A = 0;\ B = 0;\ \dot{C} = 10\ °/s;\ \dot{X} = 0.3\ m/s;\ \dot{Y} = 0.3\ m/s$
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Not Applied
Controller Specification	PD Controller for A, B; P Controller for $\dot{C}, \dot{X}, \dot{Y}$.



Result 1: Dynamics of the States

Here e is the error between the references and the states:

$$e = x_r - x \tag{4.22}$$

Giving that the objective and the corresponding system input set is $L = [x_r \quad u_r]^T$. If L result in the balance of the system, we shall call L_e an equilibrium objective. Under all equilibrium objectives, the final control outcome will be $e = 0$. This indicates that the system is well-balanced at the objective states.

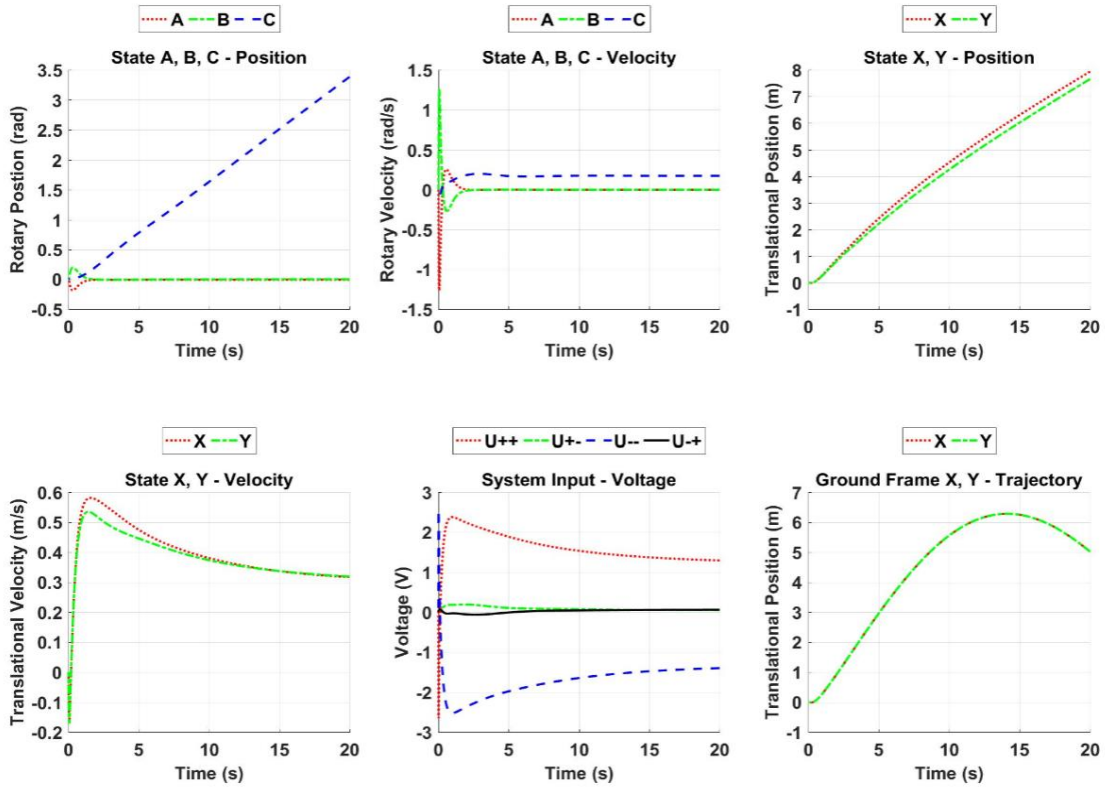
However, when L is not equilibrium objective, indicating x_{obj} and u_{obj} will not lead to the balance of the system, the result $e = x_{obj} - x \neq 0$. The system may still be statically stable. However, it will converge to another equilibrium point. For an MIMO system like Q-Baller, such problem is common since the system inputs are in most cases affecting multiple states, as shown below in Exp.4.1

From the result of Exp. 4.1, we can easily realize that the result converged to an equilibrium point which is not the objective. After we changed the control strategy, the effect of PI controller is shown in Exp. 4.2.

As we can see, the velocity control is accurate this time, which proves the idea in the second deduction. Overshoots in the velocity state plots are expected from the effect of PI controllers. These overshoots may lead to instability of the system.

To adjust the overshoots to moderate extents, we proposed a controller named "$P_{0.5}I$" velocity controller especially for X and Y. Through observation, we discovered that K_{ZP} should not be modified since it is crucial to the stability of the system.

Experiment 4.2: PI Velocity Controller Simulation based on LQR Controller	
Tracking Reference	Starting from the zero point; Track $A = 0;\ B = 0;\ \dot{C} = 10\ °/s;\ \dot{X} = 0.3\ m/s;\ \dot{Y} = 0.3\ m/s;$
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Not Applied
Controller Specification	PD Controller for A, B; PI Controller for $\dot{C}, \dot{X}, \dot{Y}$.



Result 1: Dynamics of the States

However, we discovered that when integral controller is applied, velocity $\dot{X}$ and $\dot{Y}$ will eventually converge to the designed reference as long as the integral error is correct, regardless of the actual velocity error applied to the controller. In this case, the PI Controller for $\dot{X}$ and $\dot{Y}$ is equivalent to the PD Controller of X and Y.

Based on this feature, for tracking references $\dot{X}_r$ and $\dot{Y}_r$, we adopted such control strategy for $\dot{X}$ and $\dot{Y}$ that the references are scaled for the velocity errors while remains the same for the position errors as shown in (4.23) and (4.24). By doing so, the controller will maintain its power to converge to stability while significantly cut down the overshoot effects.

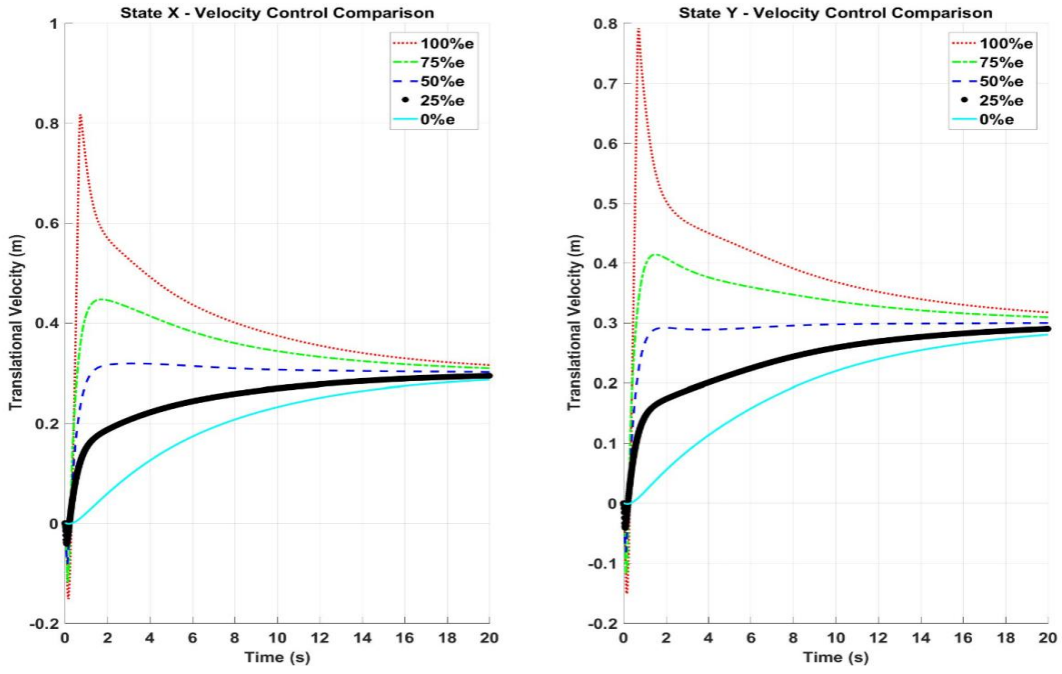
$$e_X = \int (\dot{X}_r - \dot{X})\, dt; \;\; e_Y = \int (\dot{Y}_r - \dot{Y})\, dt; \tag{4.23}$$

$$e_{\dot{X}_M} = S_X \dot{X}_r - \dot{X}; \;\; e_{\dot{Y}_M} = S_Y \dot{Y}_r - \dot{Y} \tag{4.24}$$

Here e_X and e_Y are the position error and the integral of the actual velocity error; $e_{\dot{X}_M}$ and $e_{\dot{Y}_M}$ are the modified velocity error where $\dot{X}_r$ and $\dot{Y}_r$ are scaled with S_X and S_Y.

Exp. 4.3 demonstrates the effect of the velocity reference scaler S_X and S_Y, which are the same as selected from $100\%, 75\%, 50\%, 25\%$ and 0% for each simulation, respectively. It is obvious that performance of $S_X = S_Y = 50\%$ has the best result for velocity control. Based on such result, we applied the so called "$P_{0.5}I$" velocity controller to X and Y. However, it should be noted that the technique may work well for velocity control, but it may not be the best method for position control of X and Y.

Experiment 4.3: Velocity Control Simulation with Different Velocity Reference Scaler	
Tracking Reference	Starting from the zero point; Track $A = 0;\ B = 0;\ \dot{C} = 10\ °/s;\ \dot{X} = 0.3\ m/s;\ \dot{Y} = 0.3\ m/s$
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Not Applied
Controller Specification	PD for A, B; PI for $\dot{C}, \dot{X}, \dot{Y}$ with different velocity reference Scaler;



Result 1: Velocity States Comparison

4.3.2. Implementation of Input Limiter

Input Limiter is more related to the practical property of the system. In chapter 2, we introduced the dynamics of the brushed DC motors:

$$\frac{T_{s_0}U}{U_0} - T_m = J_w\dot{\omega} + b_w\omega + \frac{60T_{s_0}}{2\pi n_{0_0}}\omega \tag{4.25}$$

Equation (4.25) is a function that omitted the motor's fast dynamics which are related to the inner electric circuitry of the DC motor, which has been mentioned in Chapter 2.

In real world, the output of the motor may be significantly affected by these factors, which will result in lagging in motor response. The maximum output of the motor is also affected by the maximum voltage input allowed by motor. The dynamic behavior of the motors may not even be linear since it may be affected by problems such as temperature rises and magnetic remanences in the motor.

An algorithm is designed for the operation of the motors named as "Input Limiter". Input limiter will be used in both simulation and real practice. While in real world application input limiter may also contain features such as overcoming motor cogging torque and voltage initial deviations. Currently at stage of the simulation, the algorithm only contains the two of the features, which are the maximum power rise limitation and the maximum output limitation respectively, as shown in Alg. 4.1:

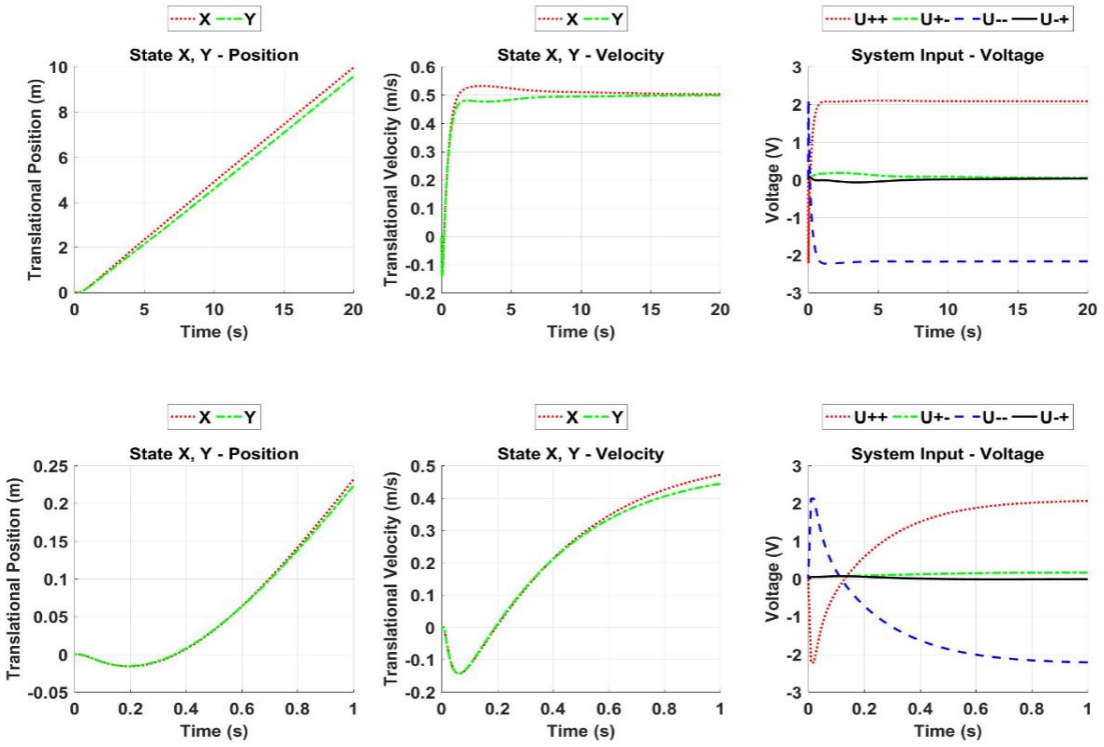
Algorithm 4.1: The Input Limiter

$\gg IF \quad (|u(t) - u(t-1)| > (h * u_{max}/t_{rise})$

$\gg \quad u(t) = u(t-1) + \Big(\big(u(t) - u(t-1)\big) * h * u_{max}\Big)/(t_{rise} * |u(t) - u(t-1)|);$

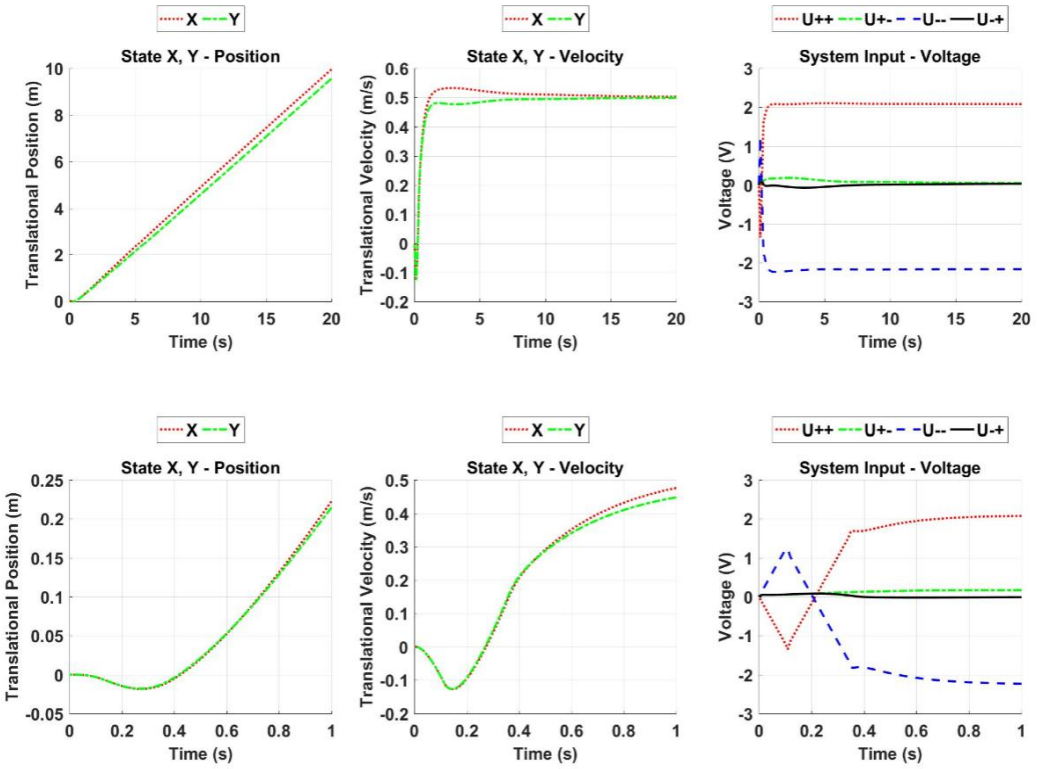
$\gg END$

Experiment 4.4: Simulation of Input Limiter	
Tracking Reference	Starting from the zero point; Track $A = 0;\ B = 0;\ \dot{C} = 10\ °/s;\ \dot{X} = 0.3\ m/s;\ \dot{Y} = 0.3\ m/s$
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Not Applied Vs. (Power Rise time = 1 s; Maximum Voltage = 11.1 V;)
Controller Specification	PD for A, B; $P_{0.5}I$ for $\dot{C}, \dot{X}, \dot{Y}$;



Result 1: Control System without Input Limiter

Continued to **Experiment 4.4:**



Result 2: Control System with Input Limiter

Continued to **Alg. 4.1**

$\gg IF \quad (|u(t) - u(t-1)| > h * u_{max}/t_{rise})$

$\gg \quad u(t) = u(t) * u_{max}/|u(t)|;$

$\gg END$

Here, t_{rise} is the required time for voltage to rise from 0V to u_{max} – the , and h is the time difference between t and $t-1$.

The effect of Input Limiter is demonstrated by the simulation of comparing the control performance of systems with and without Input Limiters respectively. As demonstrated in Exp. 4.4, the steady state performance is hardly affected by the limiter,

but the dynamics in the first second has been affected drastically. With input limiter, the system is obviously operating in a less aggressive behavior, proving the idea in the third deduction. However, it should be pointed out these algorithms will cause delay in controller, so the parameters should be selected wisely.

4.3.3. Modification of Controller for improved controller Performance

During the simulations, we also discovered a disappointment in the Yawing response quickness due to the weak controller gain according to the Yaw states. The settling time for the Yaw state would not satisfy the performance requirement. To solve this problem, we modified the parameter of the LQR controller to generate a new linear controller according to zero point:

$$Q_{LQR_{New}} = diag([100\ \ 100\ \ 10000\ \ 20\ \ 20\ \ 50\ \ 50\ \ 5000\ \ 10\ \ 10])$$

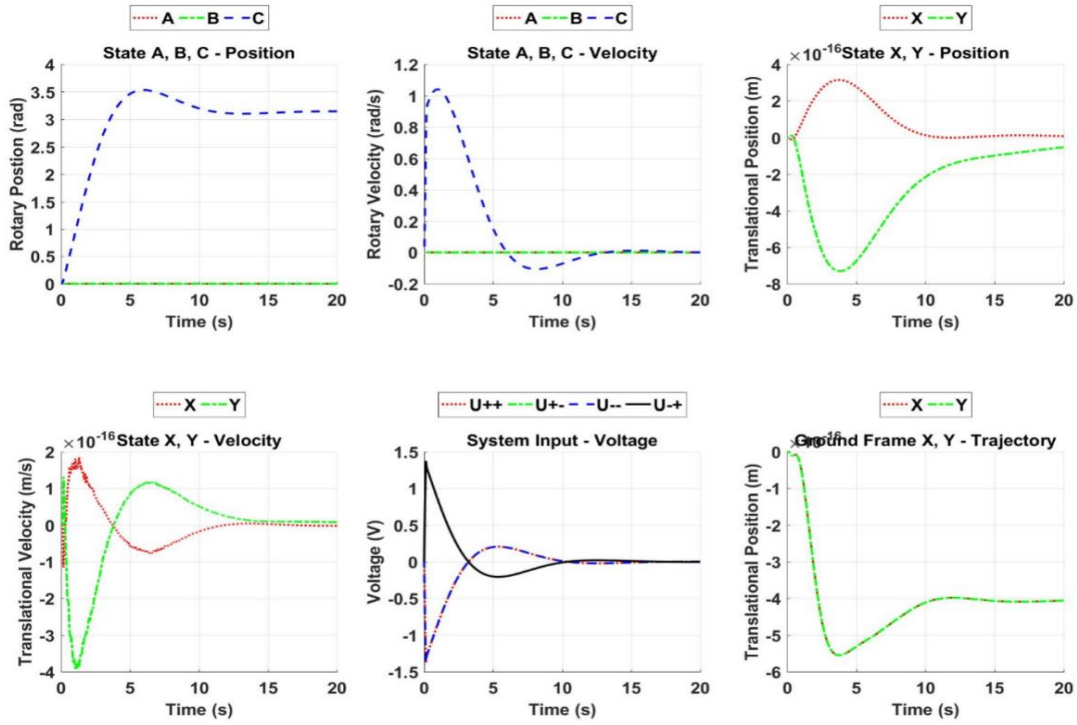
$$R_{LQR_{New}} = diag([50\ \ 50\ \ 50\ \ 50])$$

Even when the parameters for the yawing look exaggerated, the new LQR Controller is generated to be:

$$K_{New} = \begin{bmatrix} 7.381 & -7.399 & -7.0711 & -0.316 & -0.316 & 1.035 & -1.046 & -4.8897 & -4.333 & -4.333 \\ -7.381 & -7.399 & 7.0711 & -0.316 & 0.316 & -1.035 & -1.046 & 4.8897 & -4.333 & 4.333 \\ -7.381 & 7.399 & -7.0711 & 0.316 & 0.316 & -1.035 & 1.046 & -4.8897 & 4.333 & 4.333 \\ 7.381 & 7.399 & 7.0711 & 0.316 & -0.316 & 1.035 & 1.046 & 4.8897 & 4.333 & -4.333 \end{bmatrix} \quad (4.26)$$

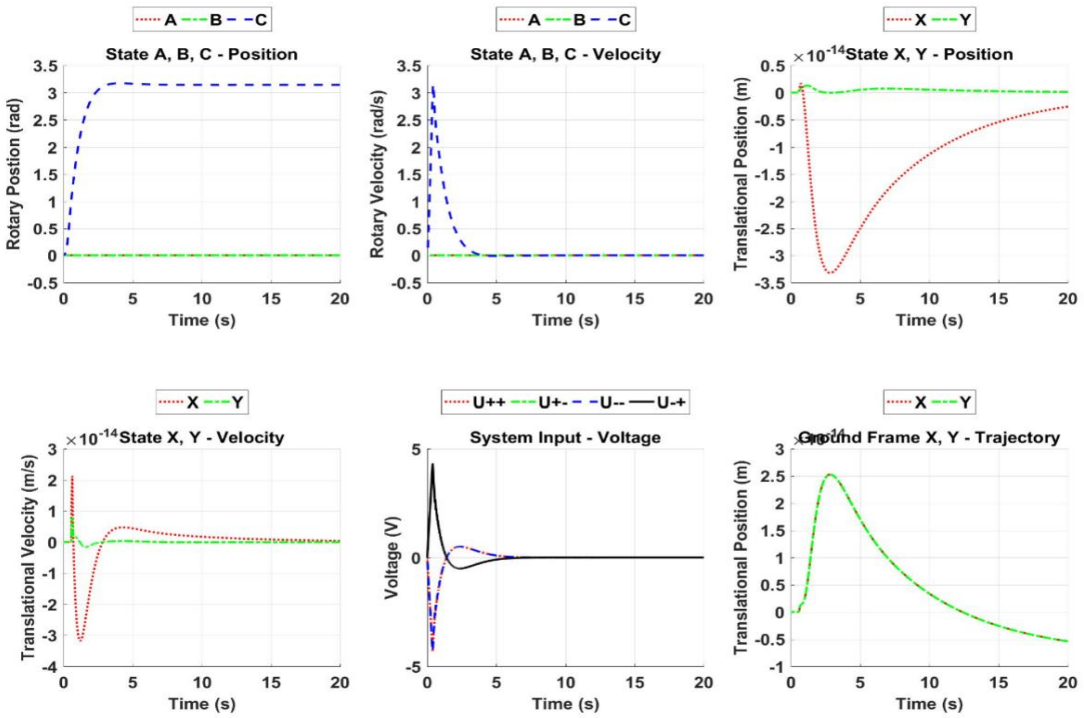
From the new controller, we can easily recognize that the controller for states other than C and $\dot{C}$ are not changed, indicating that the yawing state are irrelevant to the other states when the system is linearized at the zero point. The controller matrix elements on the 3rd and the 8th column of the matrixes are magnified by $10\sqrt{2}$ since the corresponding parameter in the Q magnified by 200 times. The new controller also satisfies the Lyapunov Stability Criterion.

Experiment 4.5: Performance Comparison between Old and New Controllers	
Tracking Reference	Starting from the zero point; Track $A = 0;\ B = 0;\ C = \pi;\ X = 0;\ Y = 0;$
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Not Applied Vs. (Power Rise time = 1 s; Maximum Voltage = 11.1 V;)
Controller Specification	Old Vs. New PD Controller for $A, B, C, X, Y;$



Result 1: Performance of Old Controller

Continued to **Experiment 4.5:**



Result 2: Performance of New Controller

Exp. 4.5 compares the performance of the old and new controllers. The new controller responses much faster by applying a much bigger input into the system. The new controller also alleviated the overshooting problem which existed in the old system. The overall performance of the controller is improved. However, we need to be careful with the error due to an increase in controller power.

4.4. Conclusion

Chapter 4 has discussed the stability and control of the system. A Linear LQR-PID Controller at the zero point which can keep the stability of the system around the zero point. Some deduction about the dynamics of the system has been proved with exemplary simulations. The linear controller's performance has also been improved through

adjustment of controller parameters and control algorithm modification. However, the linear controller cannot maintain balance when the Q-Baller's translational velocity is beyond $1\ m/s$. Nonlinear controllers will be designed in the later chapters to provide solution to the nonlinear control problem.

CHAPTER 5. GAIN SCHEDULED CONTROLLER DESIGN

The linear controller at zero point is proposed in Chapter 4.2 and improved in Chapter 4.3. The controller is very limited since it is designed based on a linearized system. Therefore, the technique of gain scheduling is adopted to solve the nonlinear control problem.

5.1. Fundamentals of Gain Scheduling

For a MIMO system without exterior disturbances:

$$\dot{x} = f(x, u) \tag{5.1}$$

$$y = g(x, u) \tag{5.2}$$

Providing the condition that the system has the following properties:

1) Continuous and differentiable in the domain $\{x, u\} \in D_{GS}$;
2) Has multiple equilibrium points $\{x_e, u_e\}$ in domain D_{GS} that satisfies:

$$\dot{x}_e = f(x_e, u_e) = 0 \tag{5.3}$$

Then the system can be linearized at the equilibrium points as:

$$\dot{X_{SS}} = \dot{X_{SS}} - \dot{X_e} = A_{SS}(x_e, u_e)(X_{SS} - X_e) + B_{SS}(x_e, u_e)(U_{SS} - U_e) \tag{5.4}$$

Giving that a feasible fixed linear controller K_S has been designed at $\{x_e, u_e\}$, the control input can be described as:

$$U_{SS} = -K(x_e, u_e)(X_{SS} - X_e) + U_e \tag{5.5}$$

The system linearized at the equilibrium point can thus be rewritten as:

$$\dot{X_{SS}} = (A_{SS}(x_e, u_e) - B_{SS}(x_e, u_e)K(x_e, u_e))(X_{SS} - X_e) \tag{5.6}$$

For different tracking reference x_r, the linearized system can be rewritten as

$$\dot{X_{SS}} = (A_{SS}(x_r, u_r) - B_{SS}(x_r, u_r)K(x_r, u_r))(X_{SS} - X_r) \tag{5.7}$$

In conclusion, the closed-loop nonlinear system can be described as:

$$\dot{x} = f(x, -K(x_r, u_r)(x - x_r) + u_r) \tag{5.8}$$

$$y = g(x, -K(x_r, u_r)(x - x_r) + u_r) \tag{5.9}$$

5.2. The Traditional Gain Scheduling

The first thing for gain-scheduling is to design the operating points. For our Q-Baller, there are 10 states in total. However, $A, B, \dot{A}$ and $\dot{B}$ should always be maintain at 0, while C, X and Y are irrelevant to the State-Space Matrix A_{SS} and B_{SS}. Of the remaining 3 states, since the control of $\dot{C}$ has better performance and robustness compared to the other velocity states, we only focus the gain scheduling on $\dot{X}$ and $\dot{Y}$ to design a more powerful translational velocity controller.

The equilibrium points are acquired through simulation experiments since it is very difficult to calculate those points directly through the governing ODEs of the system. Starting from the zero point, the system is controlled to accelerate to and maintaining balance at certain velocity.

Since K_{ZP} is not robust enough to maintain balance when the combined velocity of $\dot{X}$ and $\dot{Y}$ go beyond $v = 1\ m/s$, the gain-scheduled controller was first designed at

several low speed equilibrium points. The system then used the preliminary gain-scheduled controller to reach higher speeds and thus gradually expand the range of the controller. The final gain scheduling point design is presented below in Fig. 5.1.

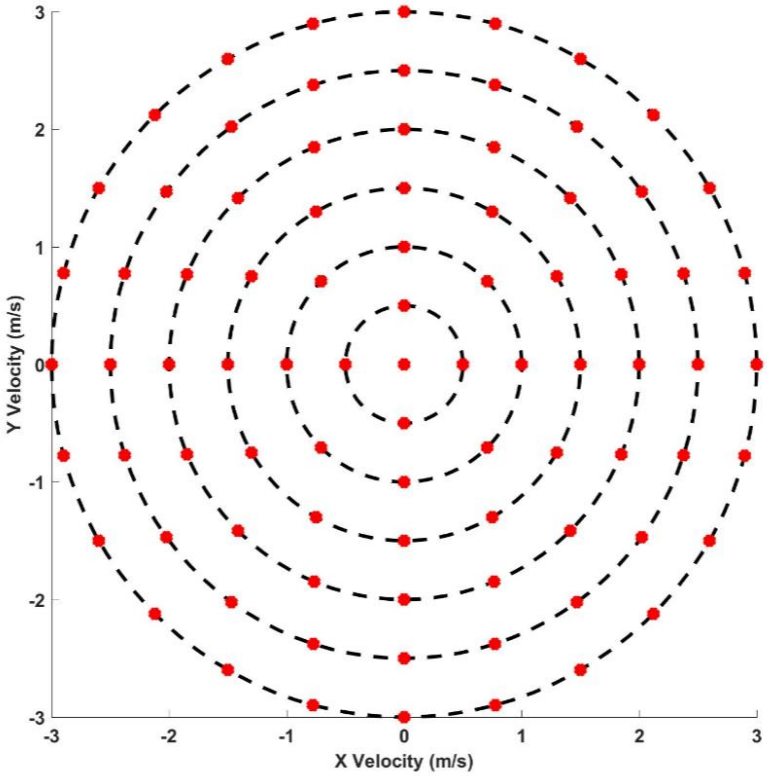


Figure 5.1: The Equilibrium Points for Gain Scheduling

There are totally 85 gain scheduling operating points, including the zero point, 4 points at $v = 0.5\ m/s$, 8 points at $v = 1\ m/s$, 12 points at $v = 1.5\ m/s$, 16 points at $v = 2\ m/s$, 20 points at $v = 2.5\ m/s$ and 20 points at $v = 3\ m/s$. LQR Controllers are designed at the equilibrium points with the parameters:

$$Q = diag([100\ \ 100\ \ 5000\ \ 20\ \ 20\ \ 50\ \ 50\ \ 3000\ \ 10\ \ 10])$$

$$R = diag([50\ \ 50\ \ 50\ \ 50])$$

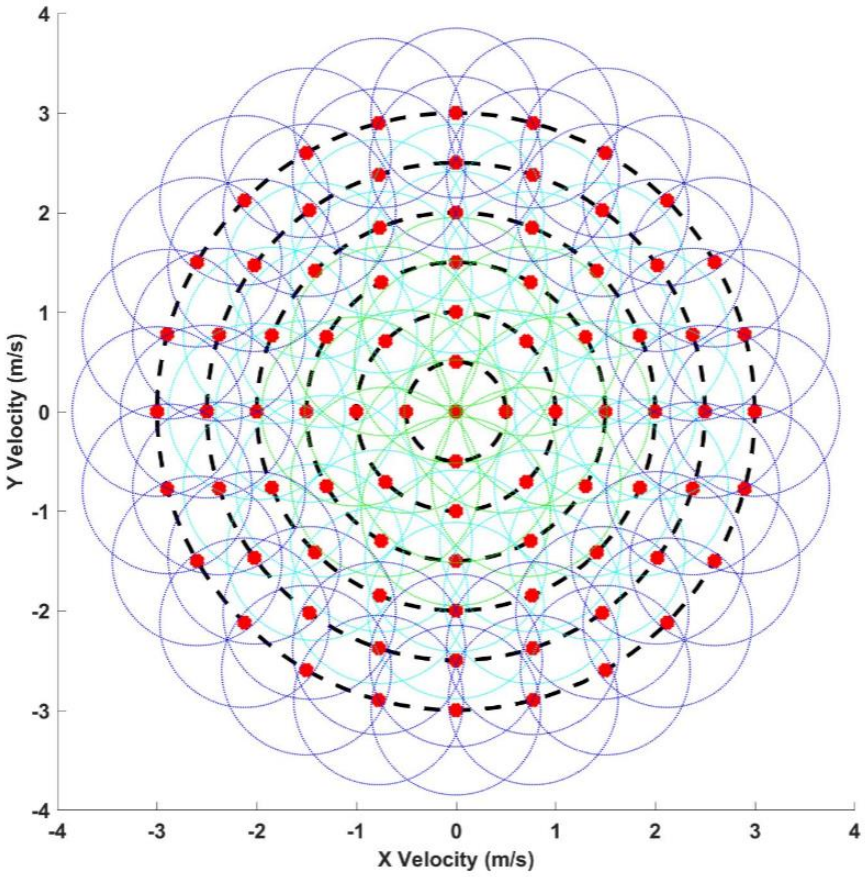


Figure 5.2: Control Domains of the Gain Scheduled Controllers

A rough depiction of the control domains of the controllers is shown in Fig. 5.2. The controller in the interior (in green and cyan colors) are generally more robust than the ones on the periphery (in deep blue color) – which indicates that Q-Baller is running on high speed with more ferocious dynamics. However, all the controllers can satisfy the gain scheduling requirement, that is, for any pair of adjoined equilibrium points p_A and p_B and their stability domain D_A and D_B, there will be:

$$p_A \in D_A;\ \ p_B \in D_B \tag{5.10}$$

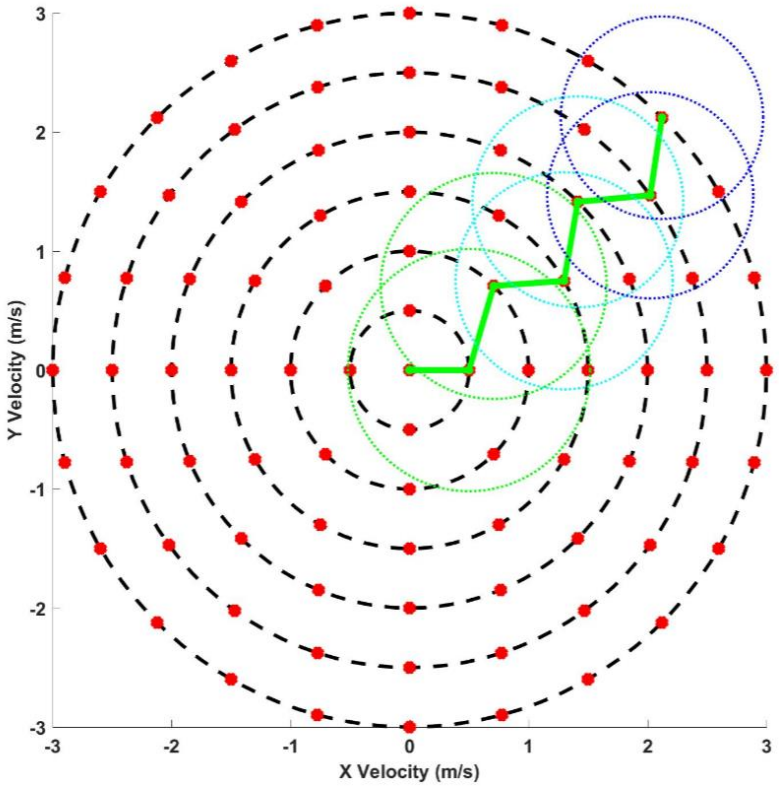


Figure 5.3: Depiction of the robustness of the controllers

When the system is tracking a reference beyond the original starting point control area, the controller of the system will switch along its way to the reference. For example, when the controller attempts to reach velocity $\dot{X} = 2.121\ m/s$ and $\dot{Y} = 2.121\ m/s$, the system controller may switch through the path as shown in Fig. 5.3. There may be other possible paths, which is mainly depended on the designed switching judgement and the states of the system at that moment.

5.3. Continuous Gain Scheduling based on Delaunay Triangulation

The traditional gain-scheduled controller introduced in the previous subchapter is effective. However, it also has its deficiencies:

1) Like all gain-scheduled controller, the translation from an operating point to another is usually not smooth due to sudden changes of controller properties. Ramping of the property variation is usually applied to alleviate the problem. However, the algorithm for ramping is usually very complicated for multi-dimensional gain scheduling regarding different changing speed for different equilibrium states.
2) The controller performance may not be the best when the reference point is not the operating point, especially when the tracking point is in between the switching point between two operating points. The controller may have very poor performance and there may also be oscillation when the control algorithm is not designed perfectly.

The general idea of overcoming these problems is simple – make the variation of the controller continuous. But to realize this with the traditional gain-scheduled controller, it requires a huge amount of operating points in the control domain, which is usually very unpractical. Here, a new gain scheduling method is proposed to solve the problem with Weighting Technique based on Multi-Dimensional Delaunay Triangulation [26].

5.3.1. Traditional Application of Delaunay Triangulation in Gain Scheduling

Giving a set I_P of points in an Multi-Dimensional space, the Multi-Dimensional Delaunay Triangulation will create a Triangulation $Tri(I_P)$ which satisfies that:

1) No other point should be contained in the Multi-Dimensional circumsphere of any simplex in $Tri(I_P)$
2) No simplex in $Tri(I_P)$ is intersecting with other simplexes.
3) The combined volume of the simplexes is the convex hull of I_P

The idea of the Delaunay Triangulation is illustrated in Fig. 5.4, which presents the Delaunay Triangulations of random point sets in 2D and 3D respectively.

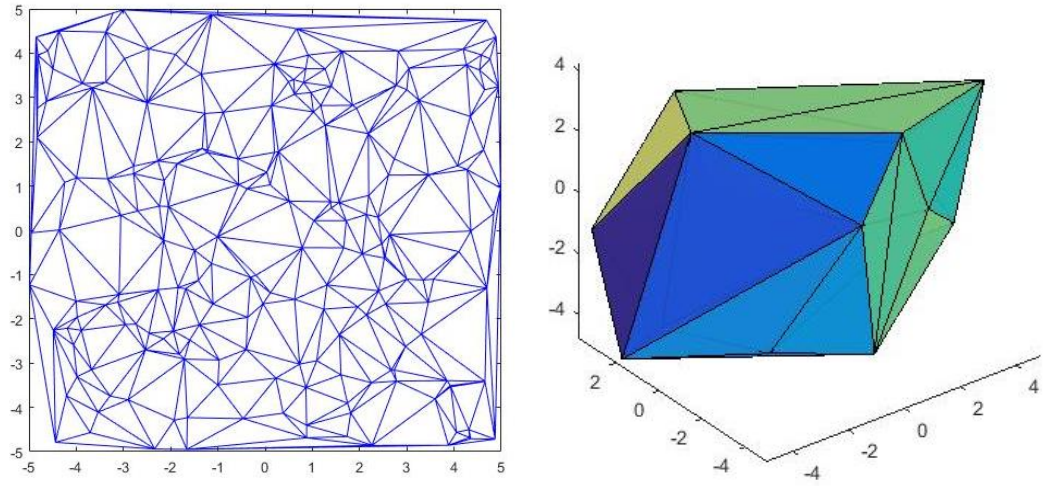


Figure 5.4: Random Point Delaunay Triangulation

(Left: 2D; Right: 3D)

The traditional Delaunay Triangulation application in Gain Scheduling requires the designer to carefully select operating points to prevent weak triangulations (when four of the points are the vertices of a rectangle). The previously selected operating points Q-Baller does not have such problem. The triangulation of the operating points for Q-Baller is shown in Fig. 5.5.

The weighting method based on triangulation uses the characteristic of the simplex, that is: for any N-Dimensional Simplex S_N whose vertices are $P_1, P_2, P_3 \ldots P_{N+1}$, any extra single point P_X in the simplex will separate the N-Dimensional simplex volume V_S into N+1 parts named $V_{S_1}, V_{S_2} \ldots V_{S_{N+1}}$. Each V_{S_n} is calculated when P_X replaces P_n. Then the Barycentric Coordinate [27][28] of P_X in the simplex S_N is defined in (5.11).

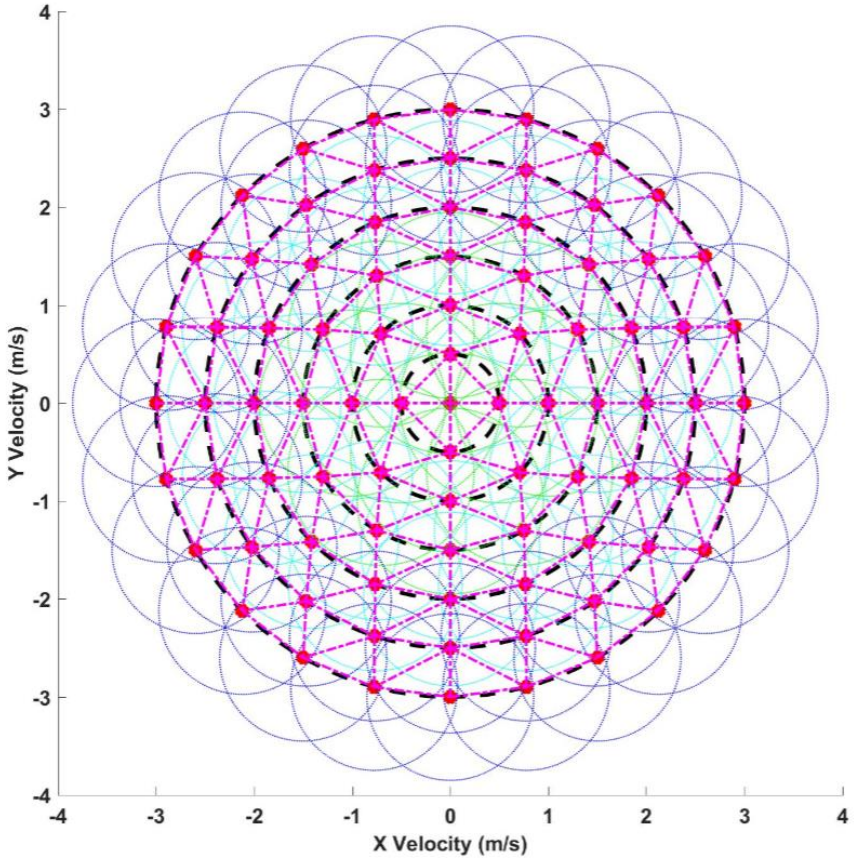


Figure 5.5: Delaunay Triangulation of the Operating Points

$$Bary(P_X) = \left[\frac{V_{S_1}}{V_S} \quad \frac{V_{S_2}}{V_S} \quad \cdots \quad \frac{V_{S_{N+1}}}{V_S}\right] = [\delta_{B_1} \quad \delta_{B_2} \quad \cdots \quad \delta_{B_{N+1}}] \tag{5.11}$$

Obviously, the Barycentric Coordinate will satisfy:

$$sum\big(Bary(P_X)\big) = 1 \tag{5.12}$$

For any weighable information N_n that the vertices P_n hold, the information N_X can then be calculated as:

$$N_X = Bary(P_X) * \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_{N+1} \end{bmatrix} \tag{5.13}$$

To apply the weighting method to our controller, for gain scheduling on a N-Dimensional space, the controller must satisfy the weighable condition which requires [29][30]:

$$A_{\mathcal{L}}(x_1, u_1) = A_{SS}(x_1, u_1) - B_{SS}(x_1, u_1)K(x_1, u_1)$$

$$\{P_S\} = \mathcal{L}\left(A_{\mathcal{L}}(x_1, u_1), x, Q_{\mathcal{L}_1}\right) \cap \mathcal{L}\left(A_{\mathcal{L}}(x_2, u_2), x, Q_{\mathcal{L}_2}\right) \cdots \mathcal{L}\left(A_{\mathcal{L}}(x_{N+1}, u_{N+1}), x, \mathrm{Q}_{\mathcal{L}_{N+1}}\right) \neq \emptyset \quad (5.14)$$

Here $Q_{\mathcal{L}_n}$ is a random symmetric positive definite matrix and $x \in D_{CR}$ which is the control region. The simplex encloses a control domain D_S must also satisfy:

$$D_S \subseteq D_{CR} \quad (5.15)$$

Then $\{P_{GS}\}$ can satisfy:

$$\{P_S\} \subseteq \mathcal{L}\left(\sum_{n=1}^{N+1} c_n A_{\mathcal{L}}(x_n, u_n), x, Q_{\mathcal{L}}\right) = \{P_{GS}\} \quad (5.16)$$

$$\{P_{GS}\} \subseteq \mathcal{L}\left(A_{\mathcal{L}}(x_1, u_1), x, Q_{\mathcal{L}_1}\right) \cup \mathcal{L}\left(A_{\mathcal{L}}(x_2, u_2), x, Q_{\mathcal{L}_2}\right) \cdots \mathcal{L}\left(A_{\mathcal{L}}(x_{N+1}, u_{N+1}), x, Q_{\mathcal{L}_{N+1}}\right) \quad (5.17)$$

Here, c_n are random positive real constants. Equation (4.35)~(4.38) can be explained as: if the controllers and operating points are well designed, the controller combined from the controllers designed at the operating points located at the vertices of a simplex can satisfy the stability requirement to control the state at any point enclosed in simplex. Therefore, the control input at x_{P_X} enclosed in S with a tracking reference x_r can be defined as:

$$\begin{aligned} u &= -K_S\left(x_{P_X}\right)\left(x_{\mathrm{P}_X} - x_r\right) + u_S\left(x_{P_X}\right) \\ &= -\sum_{n=1}^{N+1} \delta_{B_n} K(x_n, u_n)\left(x_{P_X} - x_r\right) + \sum_{n=1}^{N+1} \delta_{B_n} u_e(x_n, u_n) \end{aligned} \quad (5.18)$$

It should be point out that:

$$K_S \neq K(x_{P_X}, u_{P_X}) \text{ and } u_S(x_{P_X}) \neq u_{P_X} \tag{5.19}$$

Here u_{P_X} is the input that maintains the balance of the system at x_{P_X}, which is closely relative to the static state error of the controller. Therefore, for the velocity controllers of the gain scheduled states, integral controllers must be applied to eliminate the static state error.

For our Q-Baller, the gain scheduling is applied on a 2D space, where the 2-Simplexes are triangles. Controller of the Q-Baller at a point in the control domain is determined by the operating points at the 3 vertices of the smallest enclosing triangle of the point.

5.3.2. Operating Point Distribution based on Energy Leveling

The calculation of the enclosing simplex of the point usually requires a large amount of time if the gain scheduling points are not organized especially for high dimension gain scheduling. Therefore, a design method named as Energy Leveling Operating Point Distribution is proposed and applied to the controller.

The basic idea of the Energy Leveling Operating Point Distribution is to select the operating points on spheres of different energy levels defined as below:

$$x_{GS}^T P_{GS} x_{GS} = v_{GS}^2 \tag{5.20}$$

Here x_{GS} is a $M \times 1$ matrix containing the N gain scheduling states, P_{GS} is a $M \times M$ diagonal positive definite matrix, v_{GS} defined as the Energy Radius. The operating points of the system will be selected on the energy spheres of different Energy Radius, and

later Delaunay Triangulation will be applied to operating points on the same spherical surfaces. For our Q-Baller, as introduced before, the energy radius is selected as the combined speed of the states, and operating points are selected on different speed circles.

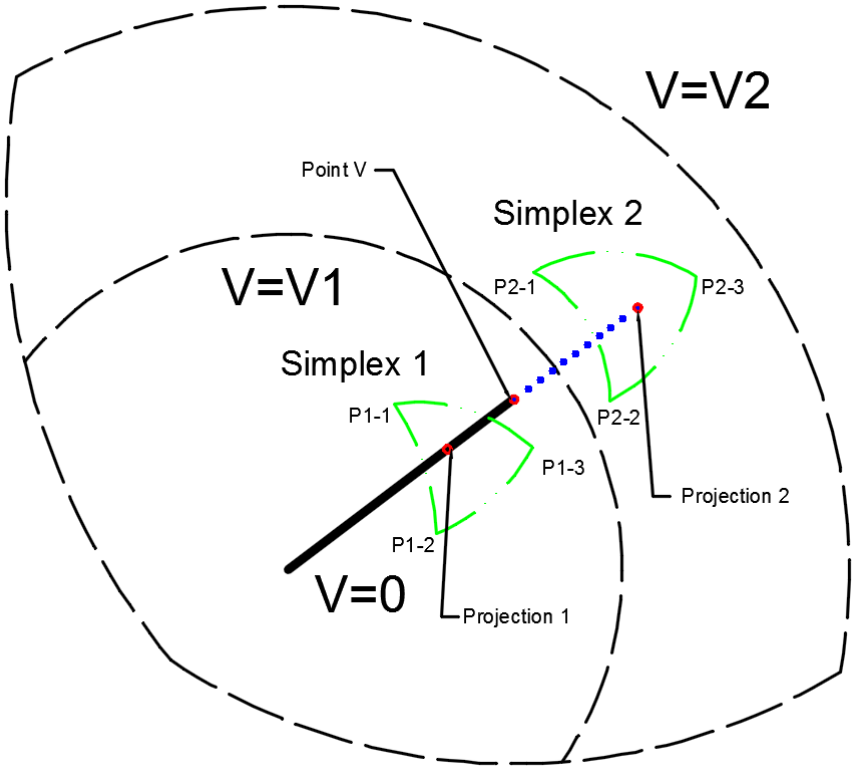


Figure 5.6: State Point projection on 3D Energy Spheres

For random points within the control domain, the points will be either on or in between energy spheres. The projections of the points along the radius onto the spheres will be enclosed by the smallest simplex from the operating points on the spherical surfaces. An example of this in a 3D gain scheduling is presented in Fig. 5.6.

To check if the projection of a point is on a certain N-Dimensional energy sphere simplex is easy, since the point that is enclosed by a sphere simplex will satisfy the following condition:

$$\begin{cases} c_n > 1 \ \ for \ \ n = 1,2,3 \dots (N+1) \\ c_1 + c_2 + c_3 + \cdots + c_{N+1} = 1 \\ \overrightarrow{OP_X} = c_1\overrightarrow{OP_1} + c_2\overrightarrow{OP_2} + c_3\overrightarrow{OP_3} + \cdots + c_{N+1}\overrightarrow{OP_{N+1}} \end{cases} \tag{5.21}$$

The operating points and the projection points are now on a spherical surface instead of a plane, and enclosing simplex is now a segment of the spherical surface. But we know that for any arc on a sphere, the arc, the center of the sphere and the chord of the arc are always on the same plane. The sphere segment can be projected on to the plane simplex made up by the chords, as illustrated by a 3D sphere example in Fig. 4.8.

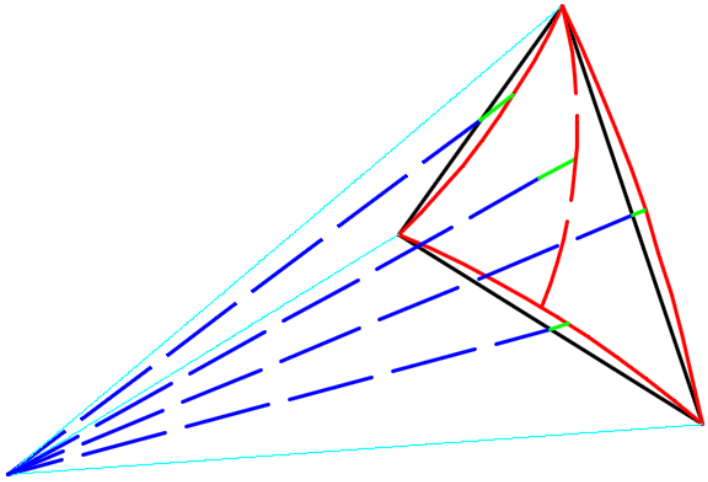


Figure 5.7: Projection of 3D Spherical Simplex to Plane Simplex

The projection of a point on a N-Dimensional sphere simplex to the spherical simplex along the radius is calculated by solving set of equation presented below [31]:

$$\begin{cases} (1). \quad C = \theta_1 u_1 + \theta_2 u_2 + \cdots + \theta_N u_N \\ (2). \quad \begin{cases} C_1 = \partial_1 u_1 + u_2 \\ C_2 = \partial_2 u_1 + u_3 \\ C_3 = \partial_3 u_1 + u_4 \\ \quad \vdots \\ C_{N-1} = \partial_{N-1} u_1 + u_N \end{cases} \end{cases} \Rightarrow P_{Proj} = [u_1 \quad u_2 \quad \cdots \quad u_N] \tag{5.22}$$

Here equation set $(4.41).(1)$ is defined by the (N-1)-Dimensional spherical simplex and $(4.41).(2)$ is determined by the radius that go through the point. Then the Barycentric coordinates can be easily calculated with P_{Proj} and the vertices of the spherical simplex.

However, when the operating points are scarcely distributed, the error between barycentric coordinates calculated from the spherical simplex may be very different from the barycentric coordinates defined on a spherical surface. This problem can be overcome by slightly increase the operating point density on the energy spheres.

For N-Dimensional Gain Scheduling, giving a random point P_V exists between two energy spheres E_{V_1} and E_{V_2}. The projection of P_V along the radius on the two spheres are P_{V_1} and P_{V_2}. The (N-1)-Simplexes S_{V_1} and S_{V_2} enclose the P_{V_1} and P_{V_2} respectively. The vertices of S_{V_n} are marked as $P_{V_{n,1}}$, $P_{V_{n,2}}$ ... $P_{V_{n,N}}$. Any weighable information N_V of P_V can therefore be calculated as:

$$W_1 = \left|\frac{V - V_2}{V_1 - V_2}\right| \text{ and } W_2 = \left|\frac{V - V_1}{V_1 - V_2}\right|$$

$$N_V = W_1 * Bary(S_{V_1}) * \begin{bmatrix} N_{V_{1,1}} \\ N_{V_{1,2}} \\ \vdots \\ N_{V_{1,N}} \end{bmatrix} + W_2 * Bary(S_{V_2}) * \begin{bmatrix} N_{V_{2,1}} \\ N_{V_{2,2}} \\ \vdots \\ N_{V_{2,N}} \end{bmatrix} \tag{5.23}$$

To perform the (4.41) in controller weighting, the two sets of operating points on the two spherical surfaces still need to satisfy (4.35) respectively. Moreover, this time the operating points also need to satisfy:

$$\{P_{S_V}\} = \{P_{S_{V_1}}\} \cap \{P_{S_{V_2}}\} \neq \emptyset \ \ when \ \ x \in D_{CR} \tag{5.24}$$

Suppose that the control domain for the active pair of (N-1)-Simplexes is D_{VP}. This time, it is D_{VP} that must satisfy:

$$D_{VP} \subseteq D_{CR} \tag{5.25}$$

Then $\{P_{GS}\}$ can satisfy:

$$\{P_{S_V}\} \subseteq \mathcal{L}\left(\sum_{n=1}^{N} c_{1,n} A_{\mathcal{L}}\left(x_{V_{1,n}}, x_{V_{1,n}}\right) + c_{2,n} A_{\mathcal{L}}\left(x_{V_{2,n}}, x_{V_{2,n}}\right), x, Q_{\mathcal{L}}\right) = \{P_{GS}\} \tag{5.26}$$

If the controller and the operating points are well designed to satisfy the stability, the control input can be rewritten as:

$$u = -\sum_{m=1}^{2}\sum_{n=1}^{N} W_m \delta_{B_{m,n}} K\left(x_{m,n}, u_{m,n}\right)\left(x_{P_X} - x_r\right) + \sum_{m=1}^{2}\sum_{n=1}^{N} W_m \delta_{B_{m,n}} u_e\left(x_{m,n}, u_{m,n}\right) \tag{5.27}$$

5.3.3. Conclusion of Continuous Gain Scheduled Controller Design

The advantages of gain scheduling through Energy Leveling Operating Point Distribution are listed below:

1) Since the system input is closely related to the energy of the system. The operating distribution through energy leveling is more reasonable. Different controller design parameter can then be easily applied to the operating points to acquire different performance at various energy level if necessary.

   Even for traditional application triangulation weighting, distribution of operating points according to energy level will result to convenience in locating the simplex that contains the operating point.

2) The information of the controller at a random point in the control domain now comes as a combination of $2N$ adjoined controllers instead of $N+1$ controllers, making the controller combination more informative especially for the points that would be located in the weak triangulated simplexes (for example when all the operating points are located on the intersections of rectangular grids).

3) Although the control algorithm needs to calculate 2 sets of Barycentric coordinates, the total calculation is still much smaller since the calculated Barycentric coordinates are now of (N-1) Dimension instead of N Dimensions. For a N-Dimensional simplex with vertices $P_0, P_1, P_2 \dots P_N$, the volume of the simplex is calculated as:

$$\mathrm{V} = \left|\frac{1}{N!} det(P_1 - P_0, P_2 - P_0, \dots, P_N - P_0)\right| \tag{5.28}$$

Here $det(P_1 - P_0, P_2 - P_0, \dots, P_N - P_0)$ is a $n \times n$ determinant based on the distance between the vertices of the simplex.

Therefore, to calculate the Barycentric Coordinates of a N-Dimensional simplex costs $N^2/(N-1)$ times of that used to calculate the Barycentric Coordinates of a (N-1)-Dimensional simplex. Obviously, the time consummation gets significantly larger when N is larger.

The drawbacks of the application of Energy Sphere Operating Point Distribution are:

1) The requirement for controller design or the amount of the operating points are higher than the previous method.

2) The projection of operating points on the Spherical simplexes will lead to large inaccuracy in the weighting of the controllers and equilibrium inputs, especially when the operating points are scarcely located on a sphere

For our Q-Baller, when Energy Sphere Operating Point Distribution is implemented during gain scheduling. The energy spheres are 2-Dimensional, indicating that the 2D sphere simplexes are arcs on a circle. The information of the controller at any point in the control domain is determined by at most 4 operating points.

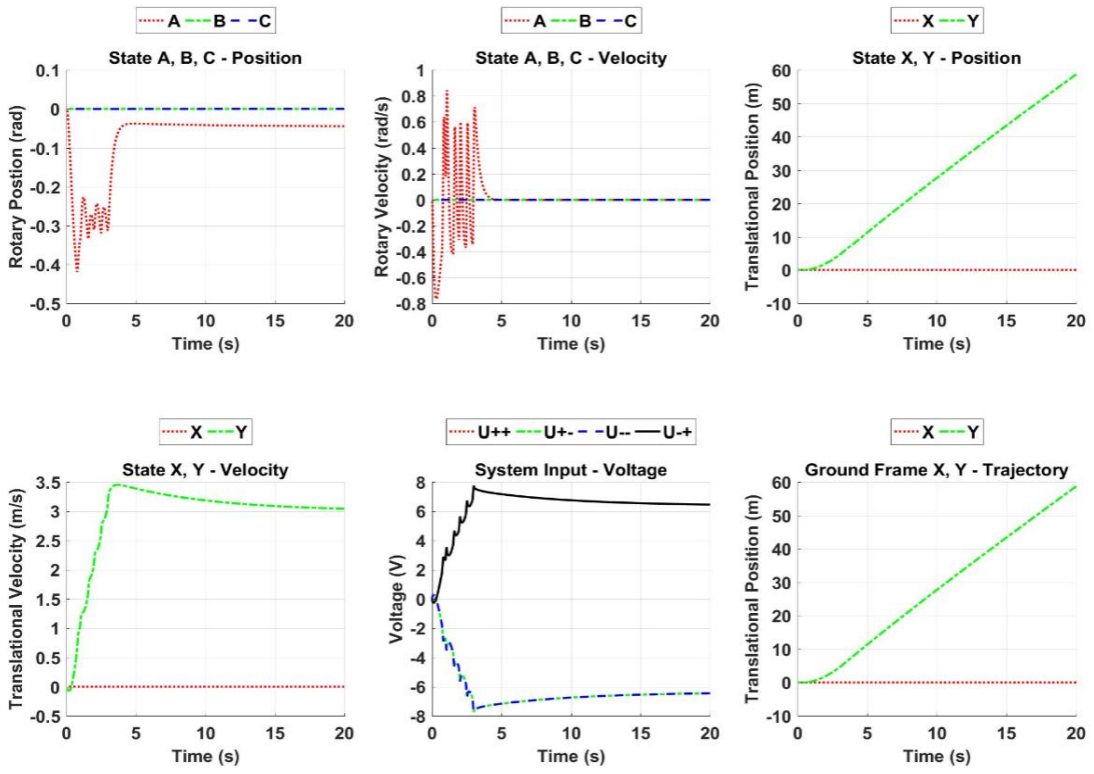
Application of both method may all lead to approximation error in initial inputs described in (5.19). While the approximation error is inevitable, Integral controllers designed for velocity controls can compensate the erroneous initial input.

5.4. Performance Comparison of the Gain-Scheduled Controllers

The three gain-scheduled controllers introduced before are compared through 2 experiments in this subchapter. The controllers are marked as GS Controller (Gain-Scheduled Controller), CGS Controller (Continuous Gain-Scheduled Controller) and EL-CGS Controller (Energy-Leveled Continuous Gain-Scheduled Controller) respectively for convenience.

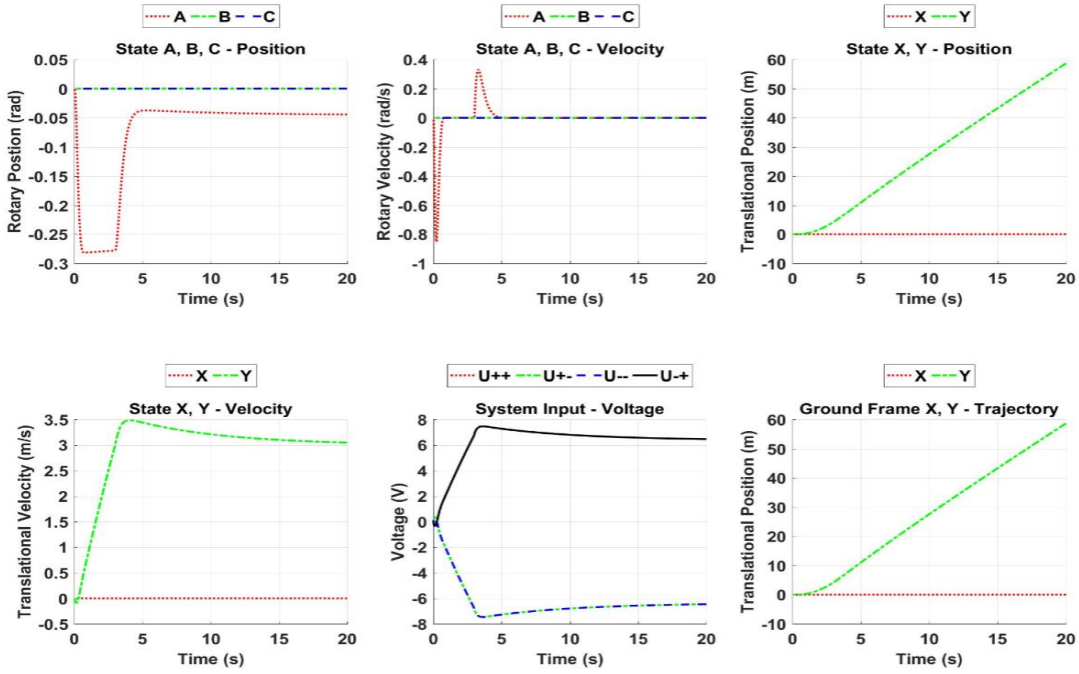
When the system is controlled with the original zero point Linear Controllers, it cannot stability beyond $1\ m/s$. Therefore, in first experiment the system is controlled to accelerate along Y direction from the zero point to $3\ m/s$ at an acceleration of approximately $1\ m/s^2$. The performances of all three of the controllers are simulated in Exp. 5.1.

Experiment 5.1: Performance Comparison between GS, CGS and EL-CGS Controllers	
Tracking Reference	Starting from the zero point; Track $A = 0;\ B = 0;\ \dot{C} = 0;\ \dot{X} = 3\ m/s;\ \dot{Y} = 0;$
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	GS, CGS and EL-CGS Controllers; Ramped PD for A, B; Ramped PI for $\dot{C}, \dot{X}, \dot{Y};\ \ddot{Y} = 1\ m/s^2$

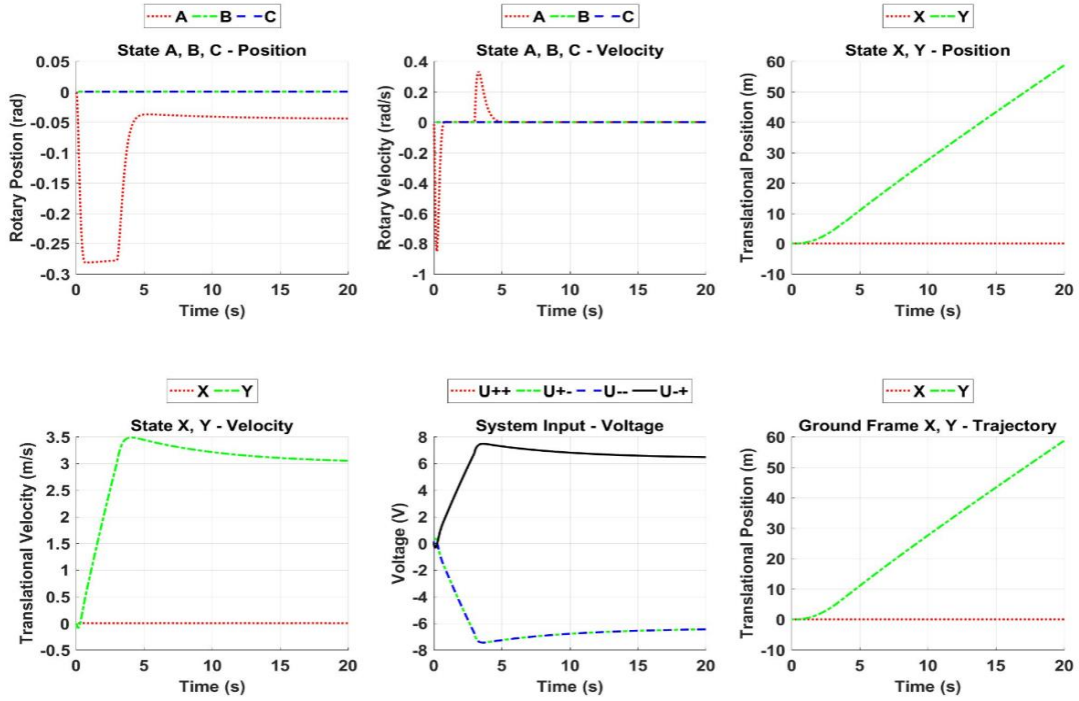


Result 1: Performance of GS Controller

Continued to **Experiment 5.1:**



Result 2: Performance of CGS Controller

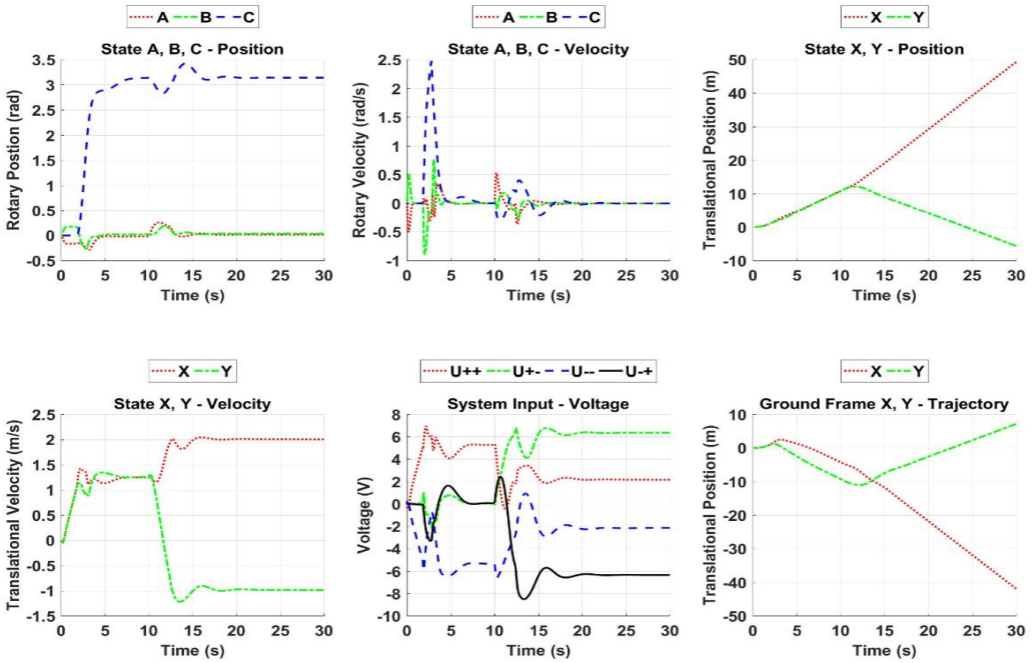


Result 3: Performance of EL-CGS Controller

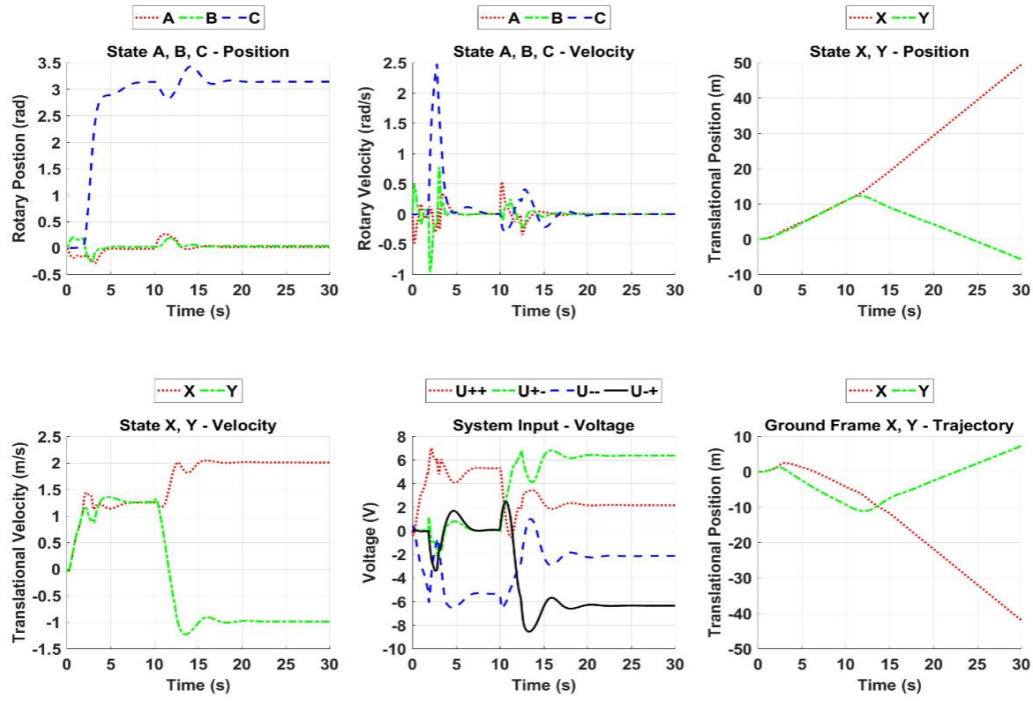
The second experiment only compared the performance of CGS Controller and EL-CGS Controller of handling multiple velocity control in sequences. The system was commanded to follow a more complicated trajectory. To eliminate the overshoot, we have applied $P_{0.9}I$ Controller for the velocity controllers of $\dot{X}$ and $\dot{Y}$ as introduced before in Section 4.3.1. The result is presented in Exp. 5.2.

Experiment 5.2: Performance Difference between CGS and EL-CGS Controllers	
Tracking Reference	Starting from the zero point: 1) Accelerate to $1.273\ m/s$ on X and Y directions at a combined acceleration of $1\ m/s^2$; (From $t = 0\ to\ t = 1.8$ second) 2) Rotate along C direction for 180 degrees in approximately 1 second. (From $t = 1.8\ to\ t = 2.8$ second) 3) Accelerate to $2\ m/s$ on X direction and -1 m/s on Y direction at a combined acceleration of $1\ m/s^2$; (From $t = 5$ second to the end of simulation)
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	CGS and EL-CGS Controllers; Ramped PD for A, B; Ramped PI for $\dot{C}, \dot{X}, \dot{Y}$; $\ddot{Y} = 1\ m/s^2$

Continued to **Experiment 5.2:**

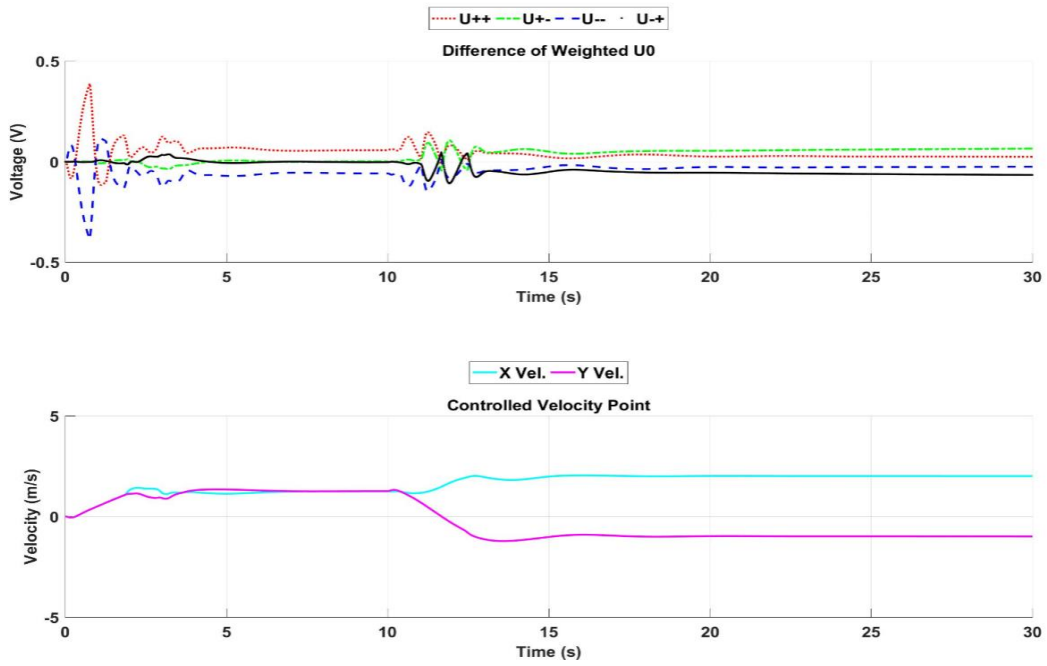


Result 1: Performance of CGS Controller



Result 2: Performance of EL-CGS Controller

Continued to **Experiment 5.2:**



Result 3: Difference in Initial Input between CGS and EL-CGS Contollers

From the results, it can be easily realized that the controller performances are very close. Both controller can keep the system stable and response fast in the control domain. However, it you pay close attention to the angular velocity performance you can see ripples in $\dot{A}$ and $\dot{B}$ the result of EL-CGS expriment.

The ripples in the plot occurs because of the inaccuracy of weighting for spherical projection described at the end of Chapter 4.4.4. The controller K_{GS} are not seriously affected, but the generated equilibrium inputs u_e are significantly different from the algorithm especially when the velocity is close to $v = 0.5\ m/s$ circle, where it only has 4 operating points on the energy sphere, as shown in Result. 3 of Exp. 4.7.

The EL-CGS controller has a slightly weaker performance compared to CGS controller in Q-Baller's system control. Since gain scheduling for a system of high

complexity and nonlinearity with a large domain of control usually will result in a large set of operating points, EL-CGS should be more suitable for these systems since it has a much shorter calculation time. Regardless of the algorithm, the application of operating point distribution on energy level sphere has been proved plausible for the gain-scheduled control of Q-Baller.

5.5. Conclusion

In Chapter 5 we discussed the design and application of gain-scheduled linear controller. To improve the performance and stability of the gain-scheduled controller, weighting method through Delaunay Triangulation has been applied to make the gain-scheduling continuous. The technique of operating point distribution on energy level sphere further improved the reasonableness of the operating point selection.

Two weighting algorithms (CGS and EL-CGS) were applied to the scheduled gains to test the performance, of which EL-CGS is specially designed for the operating point distribution on energy level sphere technique. Both controller have drawbacks related to the stabilization accuracy of control parameter after weighting method is applied, but they can maintain the stability of the Q-Baller system and improve the smoothness of the dynamics behaviors.

The final controller is capable to let the robot model reach $3\ m/s$ at an acceleration of $1\ m/s^2$ in an ideal environment without uncertainty and exterior perturbation. It is also able to keep a rotary velocity of $180\ degree/s$ when the combined translational velocity is below $2\ m/s$. Such control performance is satisfying enough for the following up simulation experiments.

CHAPTER 6. TRAJECTORY FOLLOWING SIMULATION

Previous chapters introduced the preliminary dynamics and control study of Q-Baller. By the end of Chapter 5 we have already established a capable control strategy for Q-Baller. The final controller designed for Q-Baller can be summarized as below:

1) There are 49 operating points for gain scheduling, including the zero point and 16 in each of the three sets of operating points uniformly assigned on the spheres of $\sqrt{|\dot{X}^2 + \dot{Y}^2|} = 1\ m/s$, $\sqrt{|\dot{X}^2 + \dot{Y}^2|} = 2\ m/s$ and $\sqrt{|\dot{X}^2 + \dot{Y}^2|} = 3\ m/s$, respectively.
2) LQR Controllers are designed at the operating points with the parameters (Q and R matrixes) listed below:

$$Q = diag([100\ \ 100\ \ 50\ \ 20\ \ 20\ \ 50\ \ 50\ \ 25\ \ 10\ \ 10])$$

$$R = diag([50\ \ 50\ \ 50\ \ 50])$$

3) EL-CGS method is applied to form the continuous network for nonlinear system control.

Previous controller simulation test proved that the controller can satisfy the basic performance requirement of the system. In this chapter, exclusive simulations will be performed to test the capability of the controller.

6.1. Simulation Setup

To perform more realistic and detailed virtual experiments of Q-Baller, a more complex simulation loop has been carefully setup, as shown in Fig. 6.1.

In addition to the fixed time step simulation utilities, the new simulation strategy now also supports real-time control and can discretize the originally continuous control model into code loops. The simulation modules for sensors, filters and noises can make the virtual robot performances more realistic. Other features such as Human-Machine Interface (HMI) and Adaptive/Machine Learning are used to exploit the potential of the system and further improve the controller performance.

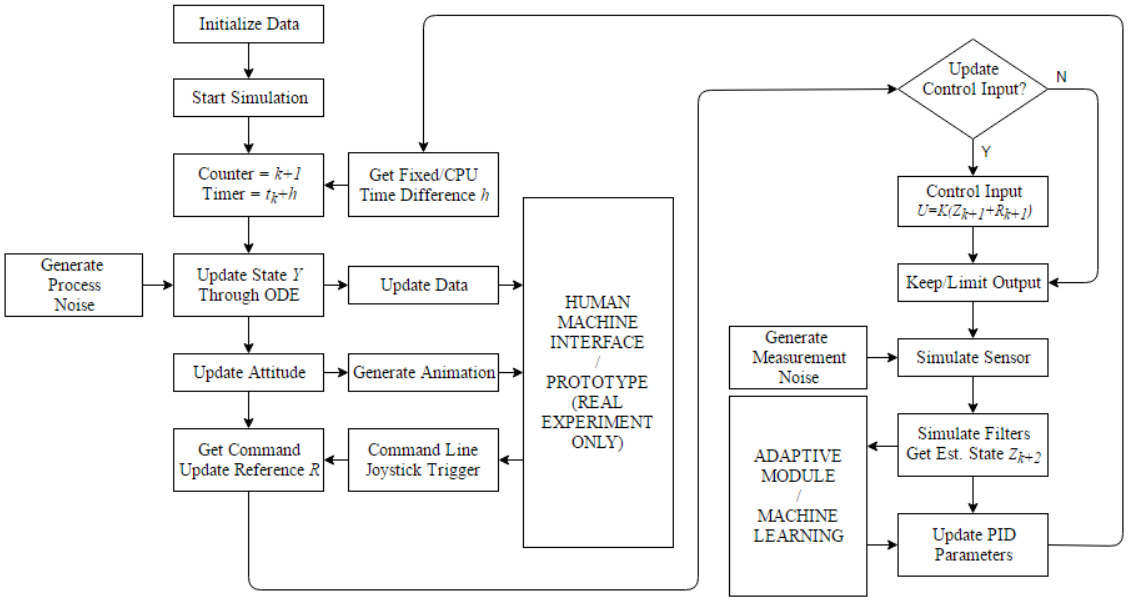


Figure 6.1: Setup of Exclusive Simulation Strategy

The idea of Human Machine Interface (HMI) is shown in Figure 6.2 [32]. The interface can generate real time plotting, 3D animation and trajectory recording. Human control input can be realized through both type-in commands and gaming joystick. The Interface is designed not only for simulation but also to serve as the control panel for future prototype experiment.

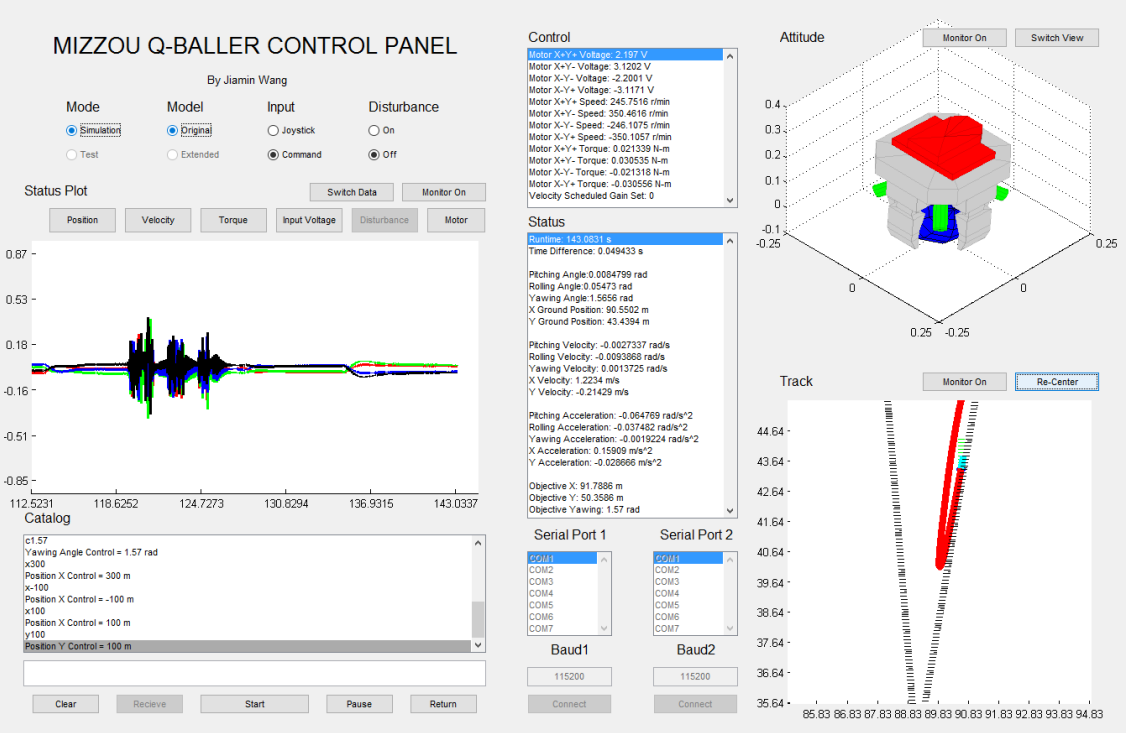


Figure 6.2: Q-Baller HMI

6.2. Reference Planning

The controller of Q-Baller only works well when the reference is provided with consideration of the characteristic of the system. Sometimes a reference far away from the current state will lead to an overwhelming error, which will lead to an impossibly large input. For Q-baller, a sudden gigantic change in input and states will result in instability of the system due to the strong coupling of the states.

Algorithms of reference planning are designed to solve the problem [33][34]. In previous simulation experiments, ramping techniques are often used to provide smooth transition of the states and inputs. The algorithms share the same idea, while it also includes other features to improve matching between the reference and the robot system.

6.2.1. Translational Velocity Reference Planning

For translational velocity control, the acceleration of velocity reference must be moderate to avoid aggressive dynamic behavior that may lead to instability. Through several controller simulation tests, we discovered that due to the low nonlinearity when the robot is closed to the zero point, the controller has stronger robustness in the zero point's neighbor domain which may allow larger acceleration of velocity reference. Therefore, we determined the relationship between acceleration and the velocity reference to be:

$$A_{rMax}(V_r) = \frac{C_1}{(|V_r|^{C_2}+1)^{C_3}} \ m/s^2, \text{ where } V_r = [\dot{X}_r \quad \dot{Y}_r \quad 0]^T \tag{6.1}$$

Here V_r is the velocity tracking reference; C_1, C_2 and C_3 are constants adjustable that can lead to the best result. You may notice that when $C_3 = 0$, the maximum reference acceleration is constant. In previous controller simulations, we tried a constant reference acceleration of $1 \ m/s^2$.

We adopted $C_1 = 1.5$, $C_2 = 2$ and $C_3 = 0.75$ which would allow a maximum acceleration of $1.5 \ m/s^2$ around the zero point and a maximum acceleration below $0.5 \ m/s^2$ when the norm of the velocity reference is close to $3 \ m/s$, which is more adaptive compared to before and can assure a more stable performance in high speed domains.

When the control system received an abrupt reference change that is vastly different from the current reference, the reference will be process by the Algorithm 6.1.

Algorithm 6.1: Translational Velocity Reference Planning

$\gg IF \quad |(V_{r1} - V_{r0})| > A_{rMax}(V_{r0}) * (t_1 - t_0)$

$\gg \quad V_{r1t} = (V_{r1} - V_{r0}) * \left(A_{rMax}(V_{r0}) * (t_1 - t_0)\right)/|(V_{r1} - V_{r0})| + V_{r0};$

$\gg \quad V_{r0} = V_{r1t};$

$\gg ELSE$

$\gg \quad V_{r1t} = V_{r1};$

$\gg \quad V_{r0} = V_{r1t};$

$\gg END$

In Algorithm 6.1, t_1 is the time of the current control iteration; t_0 is the time of the last control iteration; V_{r1} is the unplanned reference directly recorded from the latest command order; V_{r1t} is the planned reference prepared for the control input update at t_1; and V_{r0} planned reference at t_0; The algorithm will be able to avoid abrupt changes of velocity reference while maintain a satisfying performance. It will not affect the final convergence or reference or tracking performance of any well-planned velocity trajectory.

6.2.2. Translational Position Reference Planning

Translational position control, compared with velocity control, are more complicated. For Q-Baller, the position control more often carried out in Frame G rather than Frame O. When the tracker or trajectory of reference are designed in Frame G, the reference is required to be first planned in Frame G through algorithm below:

Algorithm 6.2: Translational Position Reference Planning

$\gg IF \quad |(P_{rG1} - P_{rG0})| > V_{rGMax} * (t_1 - t_0)$

$\gg \quad P_{rG1t} = (P_{rG1} - P_{rG0}) * \left(V_{rGMax} * (t_1 - t_0)\right)/|(P_{rG1} - P_{rG0})| + P_{rG0};$

$\gg \quad P_{rG0} = P_{rG1t};$

$\gg ELSE$

$\gg \quad P_{rG1t} = P_{rG1};$

$\gg \quad P_{rG0} = P_{rG1t};$

$\gg END$

Algorithm 6.2 is similar compared to Algorithm 6.1, except that the V_{GMax} can be any constant as long as it is contained in the velocity control domain. Like it Alg. 6.1, in Alg. 6.2, P_{rG1} is the unplanned ground position reference; P_{rG1t} is the planned reference prepared for the control input update at t_1; and P_{rG0} planned reference at t_0.

The position reference in Frame G is then translated into Frame O. For position trajectory tracking, the velocity profile of the trajectory is usually required to improve the effectiveness of trajectory tracking. If the position track is well designed, it is also possible to extract the detailed velocity information from the position trajectory through numerical differentiation. The algorithm of coordinate system translation and velocity profile extraction is listed below:

Algorithm 6.3: Coordinate System Translation

$\gg P_{rO1} = P_O + C_G^O(P_{rG1t} - P_G);$

$\gg V_{r1} = (P_{rO1} - P_{rO0})/(t_1 - t_0);$

In Alg. 6.3, P_{rO1} and P_{rO0} are the translational position references in Frame O at t_1 and t_0 respectively; C_G^O is the translation matrix from Frame G to Frame O as defined in Chap. 3; P_G and P_O are the latest translational position feedback in Frame G and Frame O respectively. To assure the stability of the system, Alg. 6.1 should proceed Alg. 6.3 for the velocity reference planning. The accuracy of V_{r1} can be further improved by adopting higher order forward numerical differentiation algorithm.

It need to be pointed out that even when the algorithm can extract the velocity planning information for trajectory planning, the velocity profile may not be continuous which may still lead to inaccuracy in trajectory tracking.

6.2.3. Yawing Position Reference Planning

The yawing motion of Q-Baller is different from but coupled with the translational motions. The reference planning of yawing is separated into two parts – the angle and the cycle based on the control purpose. Algorithm of Yawing Position Reference Planning is presented in Algorithm.

Algorithm 6.4: Yawing Reference Planning

$\gg P_{rYaw1} = 2 * \pi * P_{rYawCycle} + P_{rYawAngle} + P_{Yaw};$

$\gg IF \quad |(P_{rYaw1} - P_{rYaw0})| > V_{rYawMax} * (t_1 - t_0)$

Continued to **Alg. 6.4**

$\gg \quad P_{rYawDir} = (P_{rYaw1} - P_{rYaw0})/|(P_{rYaw1} - P_{rYaw0})|$

$\gg \quad P_{rYaw1t} = P_{rYawDir} * \left(V_{YawMax} * (t_1 - t_0)\right) + P_{rYaw0};$

$\gg \quad P_{rYaw0} = P_{rYaw1t};$

$\gg ELSE$

$\gg \quad P_{rYaw1t} = P_{rYaw1};$

$\gg \quad P_{rYaw0} = P_{rYaw1t};$

$\gg END$

In Alg. 6.4, $P_{rYawCycle}$ is the desire cycle of revolution for Q-Baller from its current state; $P_{rYawAngle}$ is the angle destination (from $-\pi$ to π); P_{Yaw} is the current Yawing State feedback; P_{rYaw1} is the unprocessed yawing position reference; P_{rYaw1t} and P_{rYaw0} are the processed yawing position reference at t_1 and t_0 respectively; and $V_{rYawMax}$ is the maximum yawing velocity determined by the performance requirements of Q-Baller.

6.3. Trajectory Tracking Simulation

The trajectory tracking performance based on the controller designed above is demonstrated through two sets of exemplary simulations. The first simulation tests the controller's ability of handling abrupt large reference changes based on reference planning. The second tests if the Q-Baller can follow a trajectory consisting both straight lines and curves. Neither of the trajectories designed for simulations contain any velocity profile.

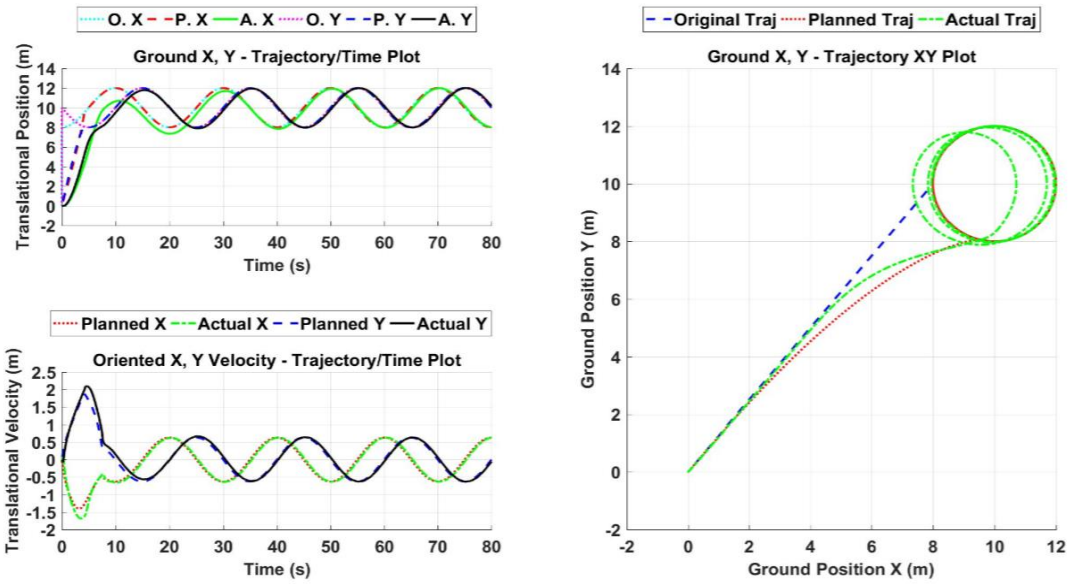
6.3.1. Trajectory Tracking of a Circle Loop

The trajectory of the first experiment is defined by the trajectory of a point looping on a circle whose center is located away from the origin of Frame G – where the Q-Baller Starts running. The Robot is commanded to follow up to the looping point. The angular velocity of the looping point is $0.1\pi\ rad/s$. The radiuses of the circles are $2\ m$, $4\ m$ and $6\ m$ respectively for each simulation.

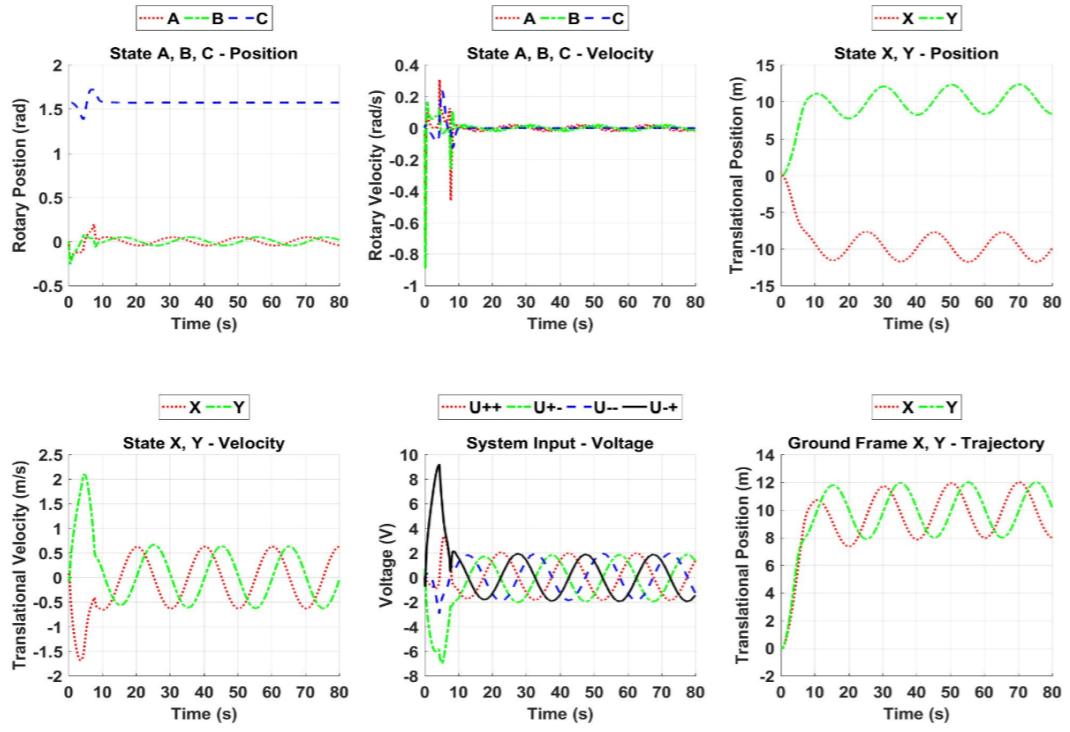
Experiment 6.1 shows the performance of the trajectory tracking of the circle $r = 2\ m$. The starting point of the tracking target was not at the Q-Baller's initial position, but the position trajectory plot in Result. 1 shows that the reference planning can improve the originally discontinuous trajectory into a smooth and continuous trajectory which eventually lead to the tracking object.

Experiment 6.1: Translational Position Trajectory Tracking of r = 2 m Circle	
Tracking Reference	Starting from the zero point; Track a moving point looping a circle of radius $r = 2\ m$ and center $O = (10, 10)$;
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	EL-CGS Controller with Translational Position Reference Planning.

Continued to **Experiment 6.1:**

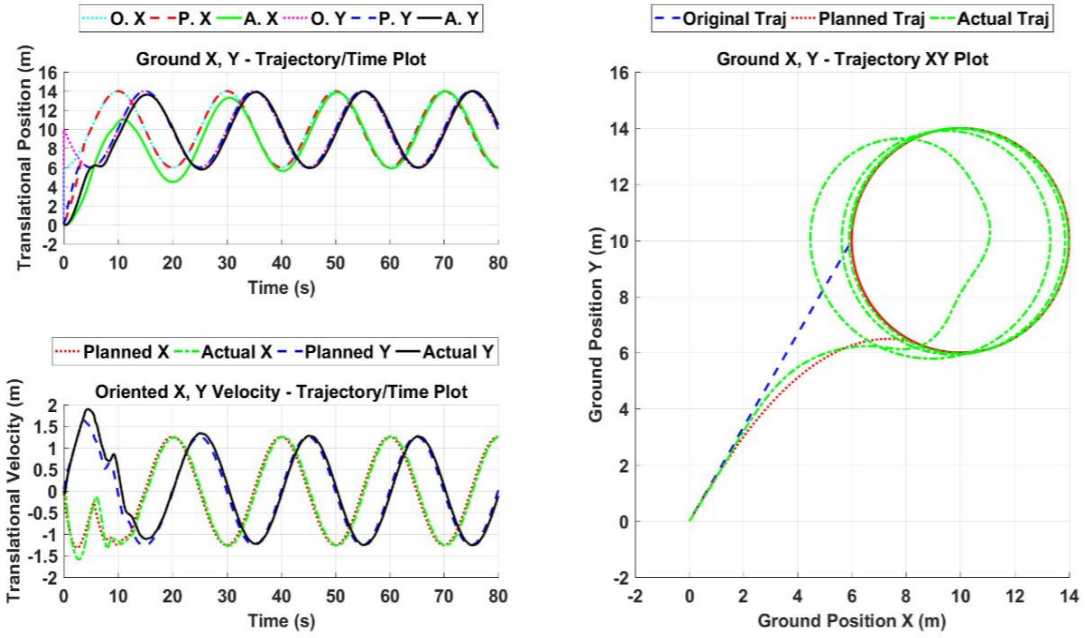


Result 1: Trajectory Tracking Performance



Result 2: State & Input via Time

Experiment 6.2: Translational Position Trajectory Tracking of r = 4 m Circle	
Tracking Reference	Starting from the zero point; Track a moving point looping a circle of radius $r = 4\ m$ and center $O = (10, 10)$;
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	EL-CGS Controller with Translational Position Reference Planning.



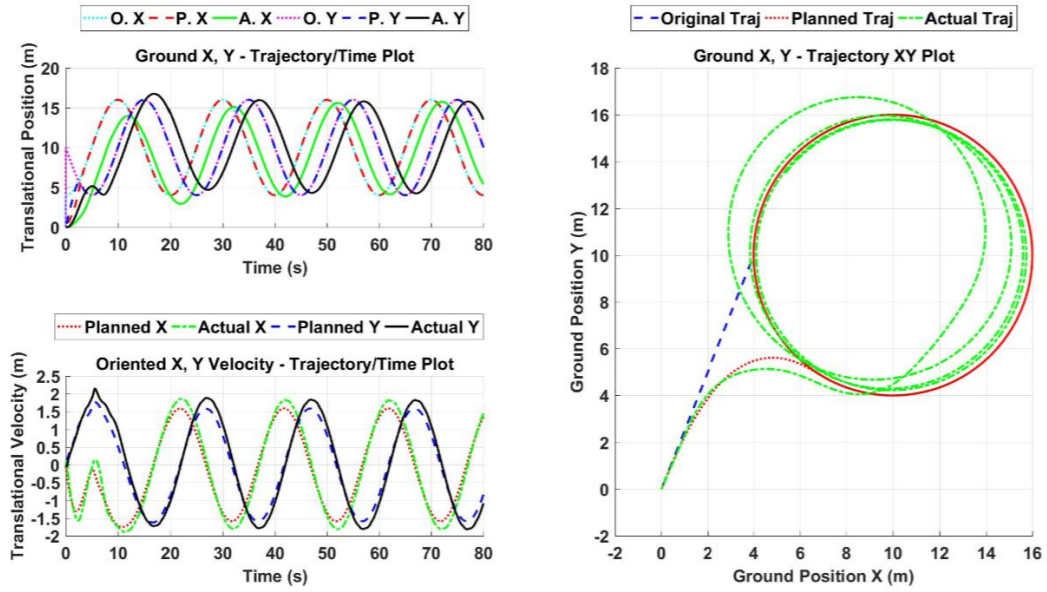
Result 1: Trajectory Tracking Performance

From the results, we learned that the Q-Baller model will converge to the planned trajectory and keep up with the looping point when the radius of the trajectory circle is $r = 2\ m$, which resulted in an average ground velocity of approximately $0.6\ m/s$. When $r = 4\ m$, the result is very similar, which is demonstrated in Experiment 6.2. The average ground velocity for the $r = 4\ m$ trajectory is approximately $1.2\ m/s$.

However, the simulation result of the $r = 6\ m$ trajectory is different from the previous two simulations. The result from Exp. 6.3 showed that the Q-Baller model was still able to keep its trajectory shape as a circle, but there exists a significant time lag between the actual position and the trajectory position. The Q-Baller model was not even able to keep itself on the $r = 6\ m$ circle.

Experiment 6.3: Translational Position Trajectory Tracking of r = 6 m Circle	
Tracking Reference	Starting from the zero point; Track a moving point looping a circle of radius $r = 6\ m$ and center $O = (10, 10)$;
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	EL-CGS Controller with Translational Position Reference Planning.

Continued to **Experiment 6.3:**



Result 1: Trajectory Tracking Performance

The reason for such poor performance comes from the reference planning. Since we put stability of the Q-Baller in the first place, the model is not able to keep up with the aggressive motion of the $r = 6\ m$ trajectory under the reference planning introduced before. The performance is expected to be much better when adopting a performance oriented reference planning strategy, while such reference planning may not be able to maintain a safe stability performance in other trajectory tracking cases.

The first set of simulations shows the general performance of reference planning. However, the trajectories designed for these simulations has been very fundamental. Therefore, we also designed a trajectory of relatively higher complexity. The result will show if the Q-Baller model can keep up with a string of connected while discontinuous motions.

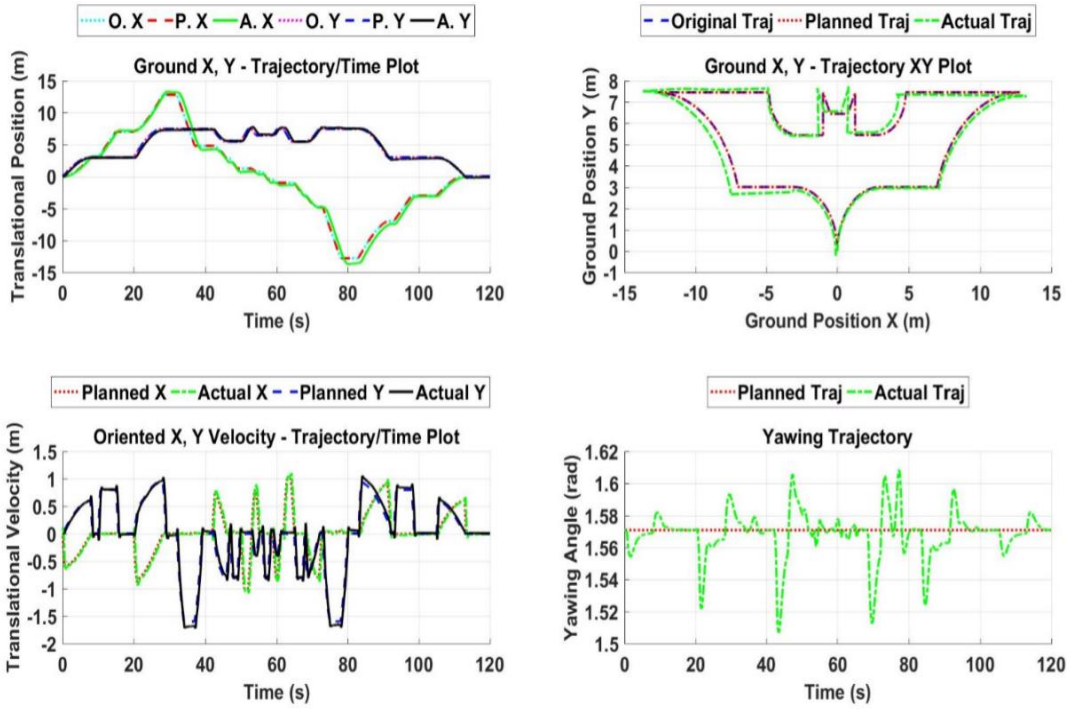
6.3.2. Trajectory Tracking of a Bat Contour

The second set of simulation test the Q-Baller Model's ability of tracking a contour whose shape looks like a bat. The Q-Baller is required to finished the contour within two minutes. The reference planning strategy is the same as before. The trajectory information and the performance of Q-Baller is presented in Exp. 6.4.

As you can see, the Q-Baller model can keep on the trajectory despite the slight position errors. The position errors occur due to the discontinuous velocity profile of trajectory design. The reference planning for position did not have much significance in this simulation case, since the position trajectory is connected and continuous in each segment. While Q-Baller is commanded to maintain the same yawing attitude, the plot in Yawing trajectory shows turbulence due to the coupling effect of the states.

Experiment 6.4: Translational Position Trajectory Tracking of a Bat Contour	
Tracking Reference	Starting from the zero point; Track a 2D Contour of a Bat Planned in 2 Minutes
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	EL-CGS Controller with Position Reference Planning.

Continued to **Experiment 6.4:**

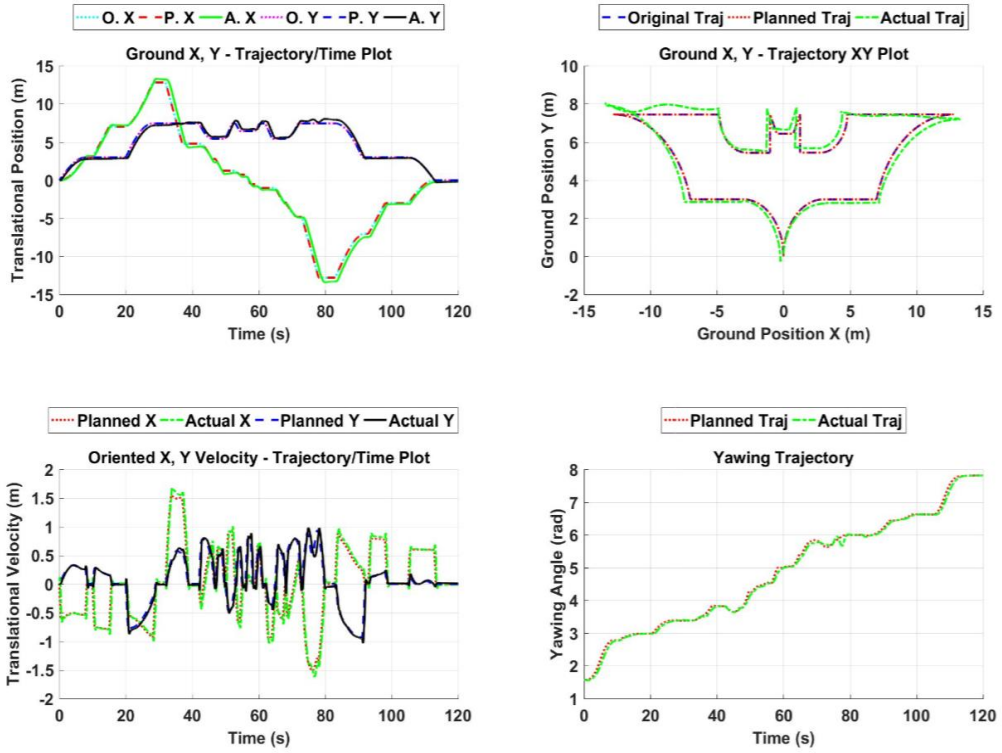


Result 1: Trajectory Tracking Performance

To test the yawing performance of the Q-Baller model, a second simulation is performed which required the model to be constantly aim its Y direction at a set point $(0,4)$ on the ground. The simulation result is shown in Exp. 6.5.

From Exp. 6.5, we learned that the Q-Baller is still capable of keeping up the pace of the trajectory and maintain the general shape of the contour despite that the coupling effect which comes from yawing of the model. The yawing control performance is also good enough for the model to keep up with the target aiming objective.

Experiment 6.5: Position Trajectory Tracking of a Bat Contour with Yawing	
Tracking Reference	Starting from (0,0); Track a 2D Contour of a Bat Planned in 2 Minutes with Y Direction Aiming at (0,4);
Noise Simulation	Not Applied
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	EL-CGS Controller with Position Reference Planning.



Result 1: Trajectory Tracking Performance

The simulation results show that the current controller design and reference planning setup of Q-Baller can realize certain level of position trajectory tracking performance. The control performance is not perfect, so there is still a lot of space left for the position trajectory tracking controller performance to be improved. However, the improvements of performances must be made based on the prerequisite that the system can maintain within the control domain.

6.3.3. Trajectory Tracking in Noisy Environment

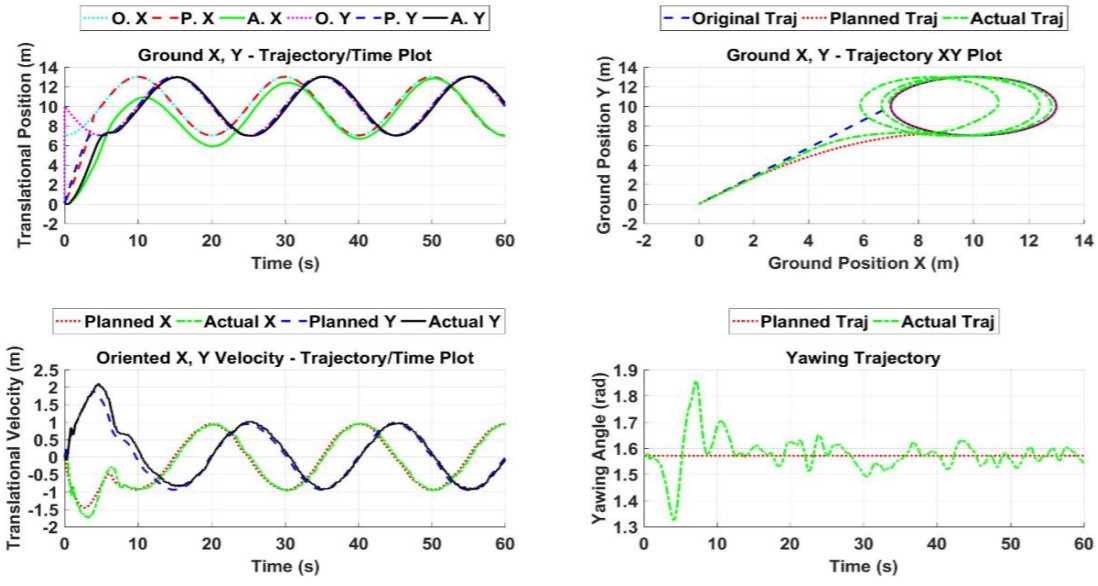
Previous simulations were conducted in ideal conditions – the model run with no process or sensor noises. Both process and sensor noises are expected in all real applications. To take a preview at the performance of Q-Baller, we attempted to simulate the noises through noise generating modules. The control performance of the system will thus be tested in a noisy environment, from which we will be able to learn whether the controller has certain robustness against noise [35].

For robust analysis and robust controller design, it is usually required to analyze the model in frequency domain. The controller designed in the frequency domain is expected to suppress certain noises of certain bandwidth while maintaining the performance of controller. At the current stage of control study, the frequency profile of noise is still undeterminable.

For noise generation at the current simulation stage, we only set the upper and lower bound of noise, and the noises were generated in uniform randomity in the time domain. The sensor feedback uncertainties of attitudes, angular velocities and translational velocities range from -1% to 1% of the actual states. Process noises are introduced as

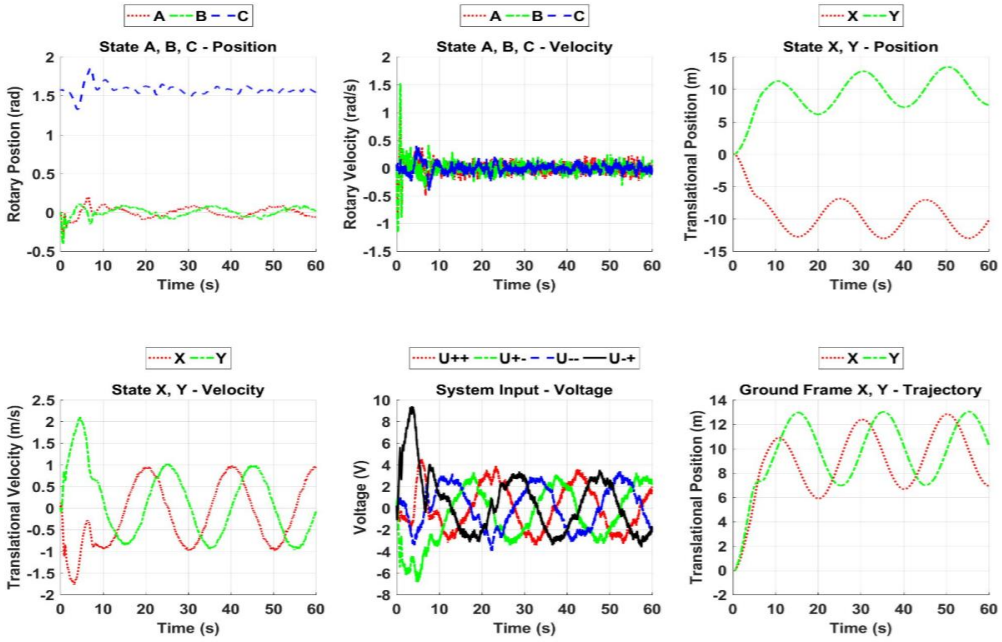
disturbance forces and torques - $\overrightarrow{T_D}, \overrightarrow{F_D}, \overrightarrow{T_d}$ and $\overrightarrow{F_d}$ as described in Chapter 3. The forces ranges from -2.5 N to 2.5 N and torques range from -0.25 N $-$ m to 0.25 N $-$ m.

Experiment 6.6: Position Trajectory Tracking of r = 3 m Circle under Noise	
Tracking Reference	Starting from the zero point; Track a moving point looping a circle of radius $r = 3$ m and center $O = (10, 10)$;
Noise Simulation	Uniform Random Distribution in Time Domain;
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	EL-CGS Controller with Position Reference Planning.



Result 1: Trajectory Tracking Performance

Continued to **Experiment 6.6:**

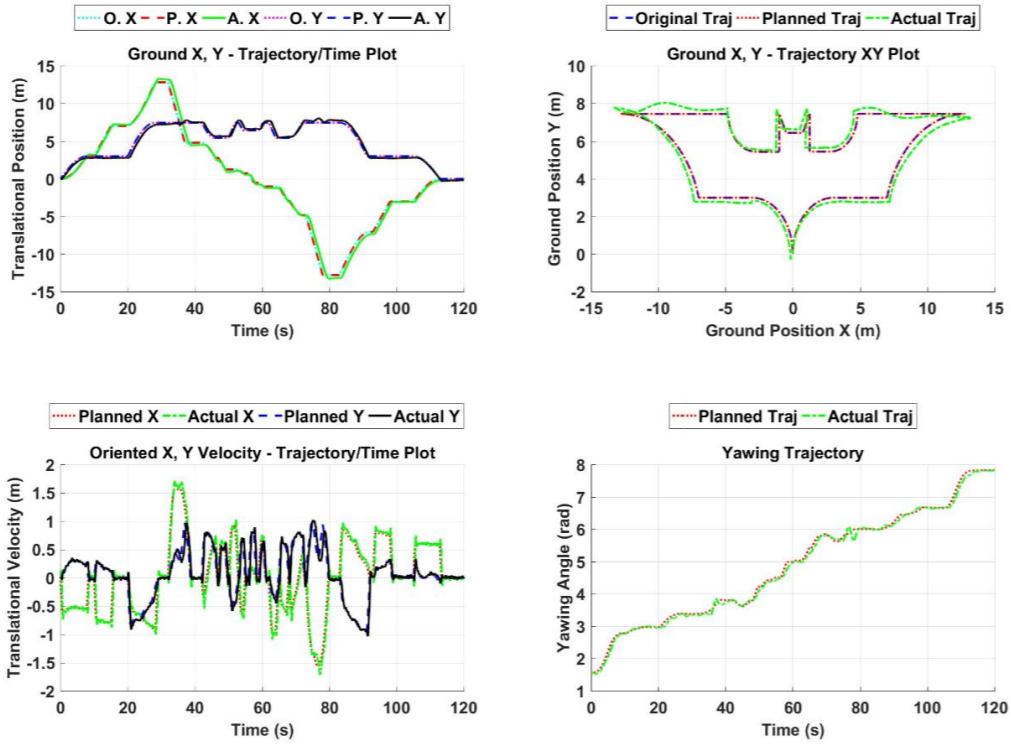


Result 2: State & Input via Time

Exp. 6.6 shows the result of Q-Baller tracking a point on the looping circle of radius $r = 3\ m$ and center $O = (10, 10)$. The plots show that the system is experiencing oscillations due to the noises. The system is still capable of maintaining the stability and tracking performance. However, when the noise amplitudes are further increased, the system will no longer be able to keep its stability. Similar conclusion can also be reached from Exp. 6.7 in which the Q-Baller model again tracks the bat contour trajectory.

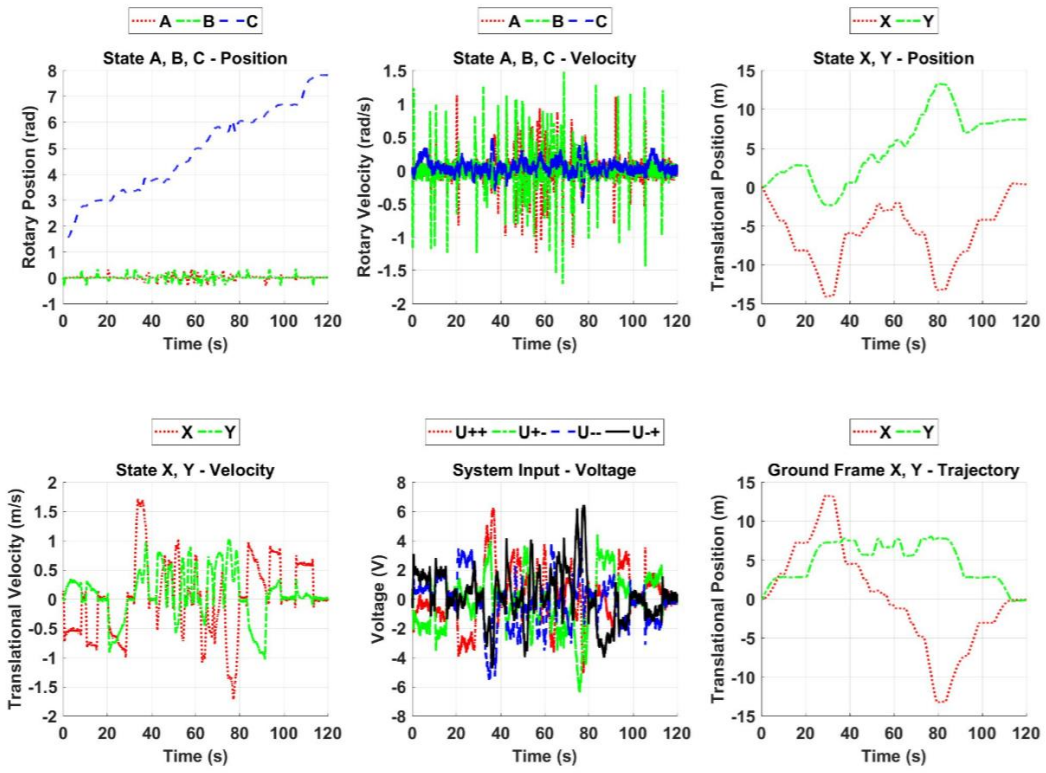
The simulation results showed that the current control system of Q-Baller has certain robustness against a certain level of uncertain noises. However, the current level of robust control study has been limited since the details about the noise for practical application is still unknown, while robust analysis should also include other aspects of study such as modeling uncertainty. Therefore, more improvements are to be made in the future.

Experiment 6.7: Position Trajectory Tracking of a Bat Contour under Noise	
Tracking Reference	Starting from $(0,0)$; Track a 2D Contour of a Bat Planned in 2 Minutes with Y Direction Aiming at $(0,4)$;
Noise Simulation	Uniform Random Distribution in Time Domain;
Observer Simulation:	Not Applied
Input Limitation	Power Rise time = 1 s; Maximum Voltage = 11.1 V
Controller Specification	EL-CGS Controller with Translational Position Reference Planning.



Result 1: Trajectory Tracking Performance

Continued to **Experiment 6.7:**



Result 2: State & Input via Time

6.4. Conclusion

In this chapter, simulations with exclusive details are conducted to study the performance of Q-Baller control system. The system model is capable of tracking trajectories of certain difficulty under noisy conditions based on reference planning. More works are expected to be done in the future to further improve the stability and performance of the system, while the current achievement through simulation study has paved the road to prototyping and real-world experiments in the future.

CHAPTER 7. PROTOTYPING & EMBEDDED SYSTEM DESIGN

The research works in the previous chapters has made the preparation for the prototyping. From the simulation results we have understood the requirement for the sensors and motors to realize the performance and stability demand. The prototyping process starts from mechanical structure to the mechatronic control system of Q-baller.

7.1. The Q-Baller Prototype

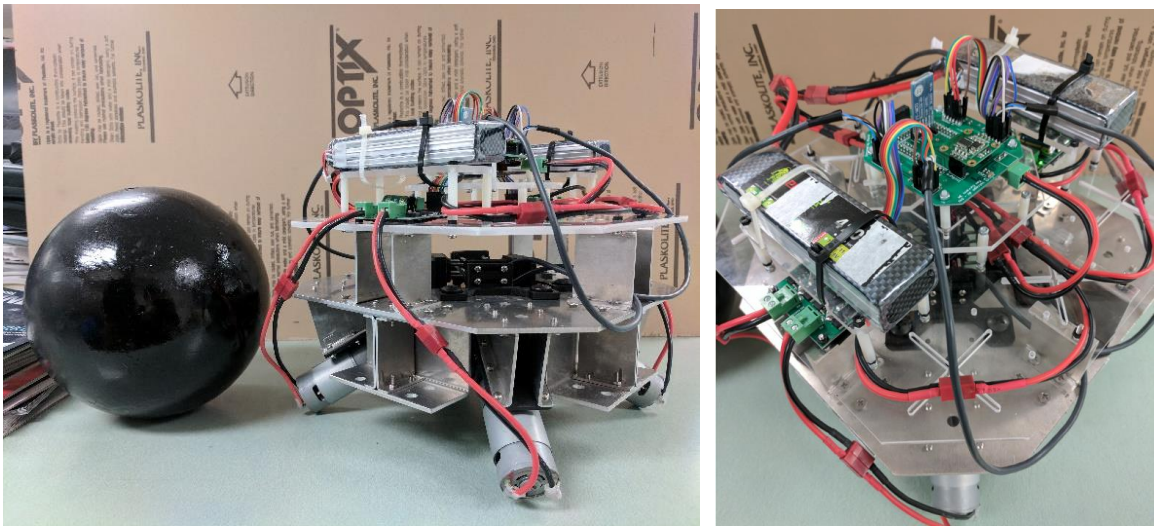


Figure 7.1: Q-Baller Prototype

Fig. 7.1 showed the complete edition of the Q-Baller Prototype. The mechatronic systems of Q-Ballers are designed as introduced in Chapter 7.1. The Q-Baller prototype system is successful in design - the mechanical and electronic components are proved functional. The prototyping process are to be discussed in this chapter.

7.1.1. Mechanical Component Manufacture & Assembling

As introduced in Chapter 2, the structural components of Q-Baller are designed into metal parts that can be manufactured simply through sheet metal forming, drilling and laser cutting. As shown in Fig. 7.2, these metal parts are manufactured in a professional sheet metal workshop with an economic cost. The components have the manufacture precisions to ensure the geometric accuracy of the components and the mechanical strength to ensure the structural safety of the robot system.

Figure 7.2: Example Mechanical Parts of Q-Baller Prototype

(Left: Sheet Metal Parts; Right: Spherical Wheel)

The spherical wheel of Q-Baller is made from a purchased steel shell coated with a layer of rubber to provide high surface traction and protecting the contacting surface from wearing out. The steel spherical shell is manufactured by casting and lathing which can provide high surface smoothness, structural rigidness and an appropriate rotary inertia.

Other structural parts are manufactured with plastic since they do not require manufacture accuracy or structural strength. Some of the plastic parts are the platform parts

on which electronic components are mounted, and therefore they cannot be metal since the conductivity of metal may cause short circuit problems if accidentally in contact with the electronic parts. Plastic parts are also lighter compared with those manufactured in common metal materials.

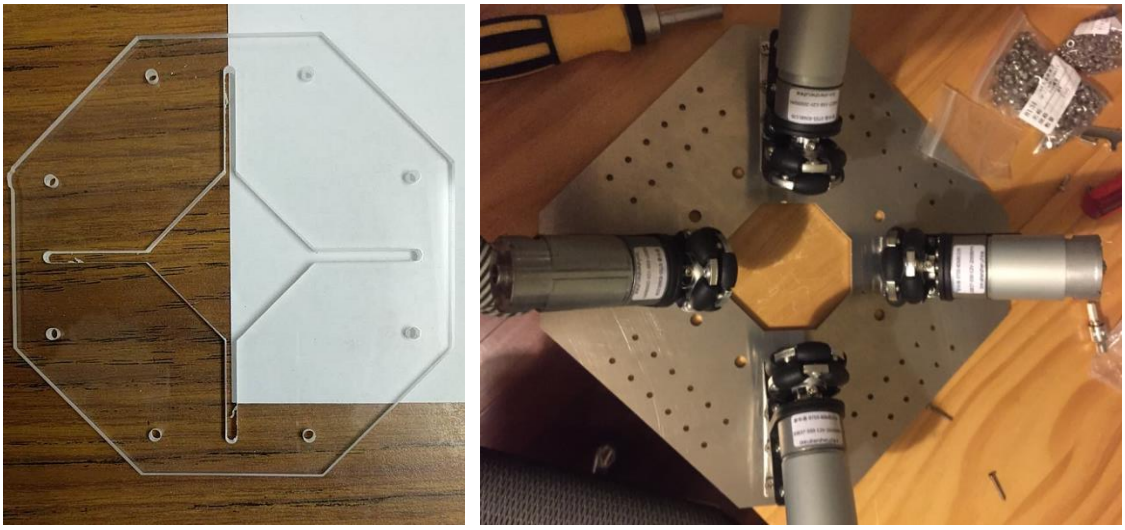


Figure 7.3: Acrylic Plastic Part (Left) and Assembling of the Prototype (Right)

Parts made in plastics can be manufactured through 3D Printing, especially the ones with complicated shapes and miniature sizes. However, for large parts with simple features, 3D Printing can be very uneconomic and time consuming. The acrylic [36] platform parts shown in Fig. 7.3 are manufactured by molding and laser engraving. The manufacture of these components is relatively coarse compared to the metal parts, but the manufacture precision is of no concern as long as they can be successfully connected with the other components are realize their functions.

The assembling of the components is realized through threaded connections. It takes around 200 pairs of bolts and nuts to connect all the components.

7.1.2. Overview of Mechatronic Control System

The electronic system of the Q-Baller has gone through multiple changes. As shown in Fig. 7.4, the preliminary electronic system made by handcraft and testing electronic circuitry board has been proved unsuccessful. The noise and disturbances between signal and power wires due to the complexity of the system has been too significant for the system to work properly. To solve the problem, a modular robotic board named "JMechW Robotic Board" (JRB) has been designed [37]. The improved electronic system has a tidier outlook which is presented in Fig.7.4.

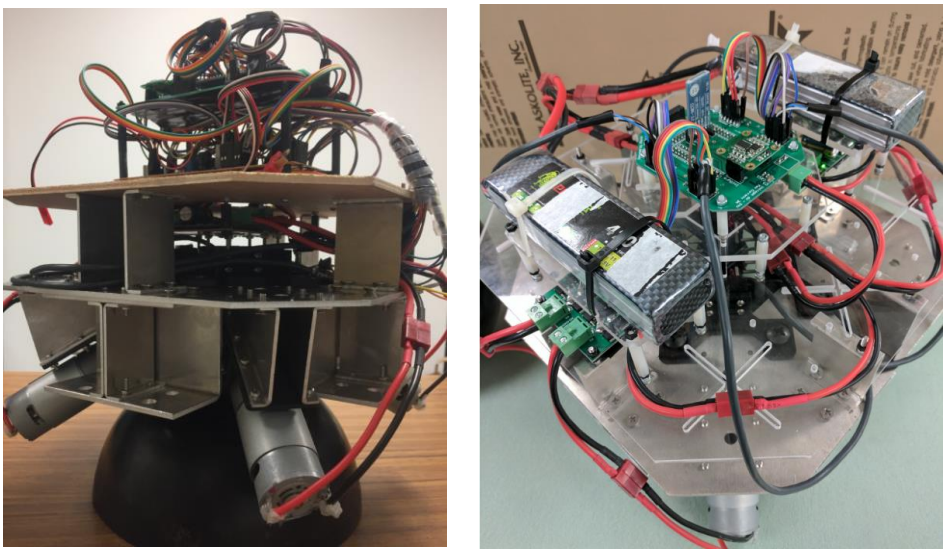


Figure 7.4: Q-Baller Prototype Mechatronic System

Left: Preliminary Electronic System; Right: Improved Electronic System

The JMechW Robotic Board has been a byproduct of the ball-bot prototype, while its application is not limited to Q-Ballers but also robotic manipulators, quadcopters or other small-scaled robotic or autonomous mechatronic systems. The circuitry board realized the modular combination of multiple sensors, controllers and output drivers.

Theoretically speaking, the circuitry can support at maximum the control of 20 DC motors or 6 stepper motors. The circuitry can also read in feedbacks from at most 2 gyroscopic sensors and 6 encoders, store data in SD card and support communication with other devices through multiple ways of high speed serial transmissions. The printed circuitry board of JRB is designed in Altium Designer and ordered for manufacture at a workshop.

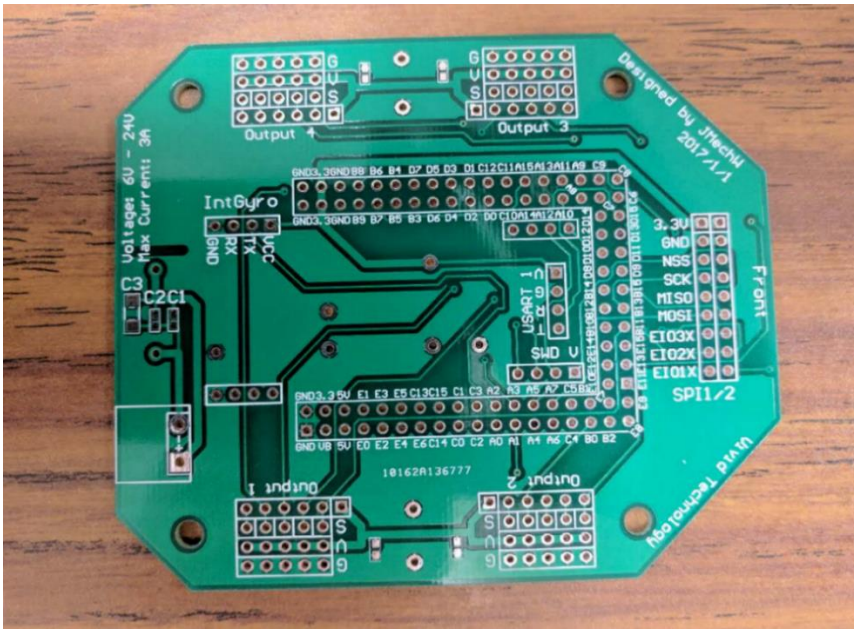


Figure 7.5: Printed Circuitry Board of JMechW Robotic Board

For the Q-Baller, the JRB board is used as the mother board to provide four ways of Pulse Width Modulation (PWM) signals for the motors, read in the feedback from a Gyroscopic Sensor and four ways of Encoders, and communicate with human-machine interface devices through a Bluetooth serial transmitter. While not all the utilities of the JRB has been utilized, the layout of the electronic control system of Q-Baller is still very complicated, which is presented in Fig. 7.6. The arrow directions in the flow chart indicate the input directions of signal or power.

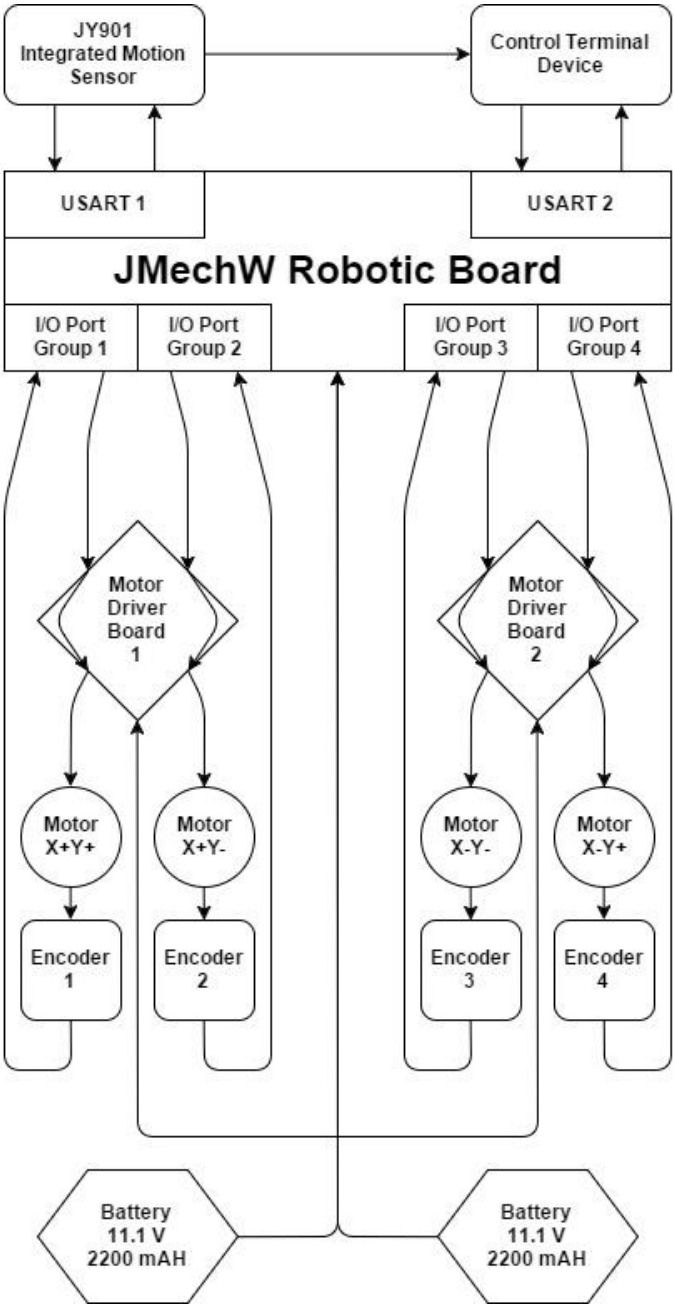


Figure 7.6: The Electronic Control System Overview

The system is powered by two batteries of 11.1V and 2200 mAH. The batteries are either parallelly or serially connected based on the requirement of the motors and motor driver boards. All microelectronic components are powered by the 5V or 3.3V power source converted from the direct high voltage from the batteries by JRB.

7.1.3. Application of Sensors

The JY901 Sensor adopted in the system is integration of gyroscopic sensor, accelerometer and compass sensor. Each sensor feedback can be used separately or combined to calculate the attitude of the Q-Baller. The feedback rate of the sensor is 100 Hz, and the data is transferred to the controller through one of the USART serial ports.

The feedbacks of gyroscopic sensor are the angular velocities around the three axes in the Local Frame of Q-Baller. The angular velocities can be integrated to achieve the Euler Angles of the system. While the integrated value may deviate from the actual value due to the accumulation of numerical calculation errors and sensor noises, the accelerometer and compass sensor can help eliminating the errors through Kalman Filtering [38].

To apply Kalman Filter to the sensors, we assume the integration of angular velocity through gyroscopic sensor as the estimator of the Kalman Filter:

$$x_{n+1,n} = Ax_n + B(u_n + v) \tag{7.1}$$

Here x_n is the feedback state at $t = n$; $x_{n+1,n}$ is the predicted state estimate at $t = n + 1$ acquired through x_n; u_n is the input of the estimator, in our cases is the angular velocity from the gyroscopic sensor; and v is the sensor noise of gyroscopic sensor.

Then it is possible to assume the pitching and rolling feedback calculated from accelerometer and the yawing direction observed through the compass sensor as the observer in the Kalman Filter system. The error between the observer and the estimator can be expressed in (7.2).

$$E = y_n - Cx_n - Du_n \tag{7.2}$$

Here in the state-space equation E is the column vector of the estimated state error at $t = n$; y_n is the observer feedback. Now it is time to proceed to the predicted error covariance estimate $P_{n+1,n}$ with P_n:

$$P_{n+1,n} = AP_nA^T + Q_n \tag{7.3}$$

The Optimal Kalman Gain K_n can then be acquired through:

$$K_n = P_{n+1,n}C^T\left(CP_{n+1,n}C^T + R_n\right)^{-1} \tag{7.4}$$

In (7.3) and (7.4), Q_n and R_n are the covariance of process and observation noises respectively. The value of Q_n and R_n can be constant or variable depending on the type of Kalman Filter. In the case of Q-Baller, Q and R are constant. The updated error covariance estimate P_{n+1} and the updated state estimate x_{n+1} are calculated in (7.5) and (7.6):

$$P_{n+1} = (I - K_nC)P_{n+1,n} \tag{7.5}$$

$$x_{n+1} = x_{n+1,n} + K_nE \tag{7.6}$$

The application concept of Kalman Filter is featured by "Predict" and "Update". The Kalman Filter for JY901 will eliminate the time lag and numerical error of attitude estimation. While a Kalman Filter Digital Signal Processing (DSP) Unit is integrated in JY901, we still tested the effect of Kalman Filter on the integration of angular velocity through a simple sensor simulation.

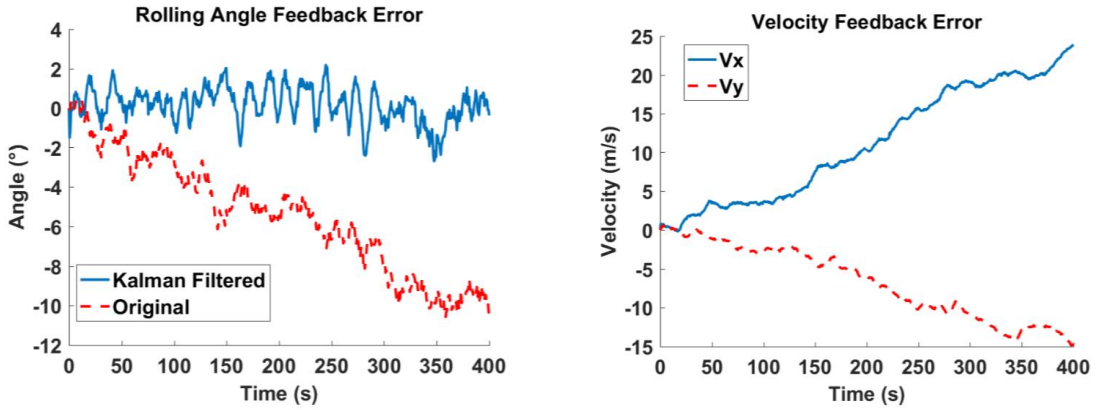


Figure 7.7: Simulation of Kalman Filter

(Left: Filtered Attitude Observation; Right: Unfiltered Accelerometer Integration)

As shown in Fig 7.7, while the actual rolling state of sensor has always been zero, the noise and integration error has led to deviation in attitude feedback. However, Kalman Filter prevented the state from deviating too far away from the actual state. As a comparison, the simulation of accelerometer has also been carried out without Kalman Filtering. The velocity observation based on acceleration integration has been unreliable, proving that the velocity and position observation of Q-Baller cannot be based on accelerometer.

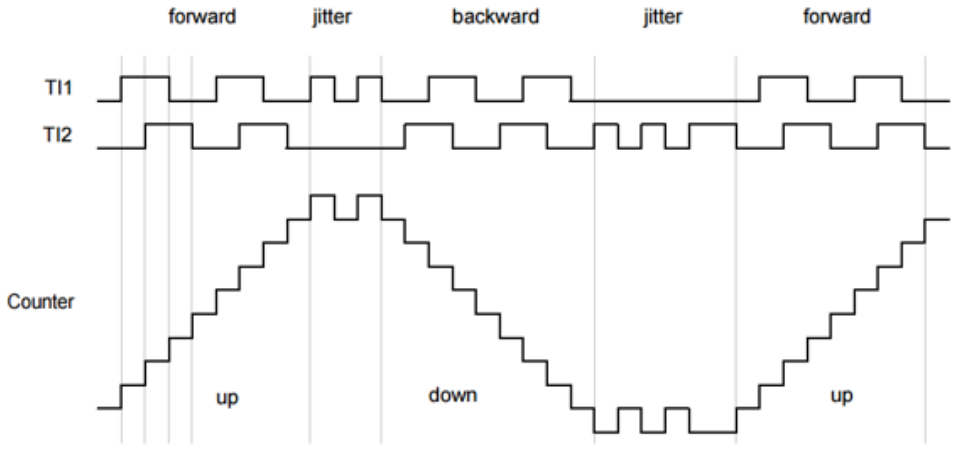


Figure 7.8: Concept of Quadrature Encoding and Decoding

This result leads to the installment of rotary encoder for the velocity and position feedback of Q-Baller. The Omron E6A2-CW3C Encoders can feedback rotary positions at the accuracy of 1/2000 and operate at a maximum speed of 5000 rpm. The position information need to be quadrature decoded from the two phases of feedbacks. The idea of quadrature encoding and decoding uses the phase difference between feedback signal to judge the direction and position of feedback, as depicted in Fig. 7.8.

After acquiring the motor rotary position data from the encoder, the velocity of motor V_{M_t} at t can be estimated with following algorithm of numerical differentiation:

Algorithm 7.1: Numerical Differentiation of Encoder Data

$$\gg h = (t_t - t_{t-3})/3;$$

$$\gg V_{M_t} = \left(\frac{11}{6}\left(P_{M_t} - P_{M_{t-1}}\right) - \frac{7}{6}\left(P_{M_{t-1}} - P_{M_{t-2}}\right) + \frac{2}{6}\left(P_{M_{t-2}} - P_{M_{t-3}}\right)\right)/h;$$

The third order numerical backward differentiation can acquire accurate motor velocity if the position plot of the motor can be considered continuous with the time step h.

Unlike numerical integration, numerical differentiation may have larger uncertainty, the feedback accuracy of the encoder has also limited the accuracy of numerical differentiation. The initial feedback frequency of encoder is selected to be 100 Hz, which may be adjusted according to performance in the future.

After that, we can acquire translational velocity $\dot{X}$ and $\dot{Y}$ in Frame O, which is presented in Alg. 7.2.

Algorithm 7.2: Acquisition of Frame O Translational Velocities

$\gg V(1:4) = [V_{M_{X+Y+}} \quad V_{M_{X+Y-}} \quad V_{M_{X-Y-}} \quad V_{M_{X-Y+}}];$

$\%This\ line\ is\ not\ used\ to\ define\ a\ variable, but\ to\ identify\ the\ elements\ of\ V\ as\ V_{M_{X+Y+}}, V_{M_{X+Y-}}\ , V_{M_{X-Y-}}\ and\ V_{M_{X-Y+}}\ in\ the\ following\ algorithm.$

$\gg V_{LX} = \frac{\sqrt{6}}{6}[-1 \quad 1 \quad 1 \quad -1]V^T \frac{r}{R};$

$\gg V_{LY} = \frac{\sqrt{6}}{6}[1 \quad 1 \quad -1 \quad -1]V^T \frac{r}{R};$

$\gg V_{LZ} = \frac{1}{2}[-1 \quad 1 \quad -1 \quad 1]V^T \frac{r}{R};$

$\gg [V_{OX} \quad V_{OY} \quad V_{OZ}] = ([V_{LX} \quad V_{LY} \quad V_{LZ}] + [\dot{A} \quad \dot{B} \quad \dot{C}])C_O^L;$

$\gg \dot{X} = R * V_{OY};$

$\gg \dot{Y} = -R * V_{OX};$

With the application of the sensors, the control system can have full access to the states of Q-Baller. The Q-Baller will be a completely observable and controllable system based on the mechatronic system. The embedded system design should make sure that all components are functional and the designed control system can be carried out smoothly.

7.2. Embedded System Development

The embedded system of Q-Baller should be designed to fulfill the design need, which is not a simple task since the control programming of Q-Baller is very complicated.

The control algorithm must be composed and compiled properly to achieve the performance as expected from the simulations. The two major parts of embedded system development are utility initiation and algorithm optimization, as the former one provides the suitable signal for the function of control system, and the latter one makes sure the control strategy can be carried out smoothly as expected in simulation.

7.2.1. Overview of STM32 Embedded System

As introduced in Chapter 2, the microprocessor we chose for the control system of Q-Baller is the STM32F407VET6 [39]. The microprocessors are not comparable to personal computers or CPU/GPU integrated robotic kits in processing speed and utility variety, but they are small sized and can be applied to build miniature but powerful electronic control systems. The processing frequency of STM32F407VET6 is 168 MHz, which is powerful enough for the basic application in Q-Baller. However, it still requires algorithm designs to work properly.

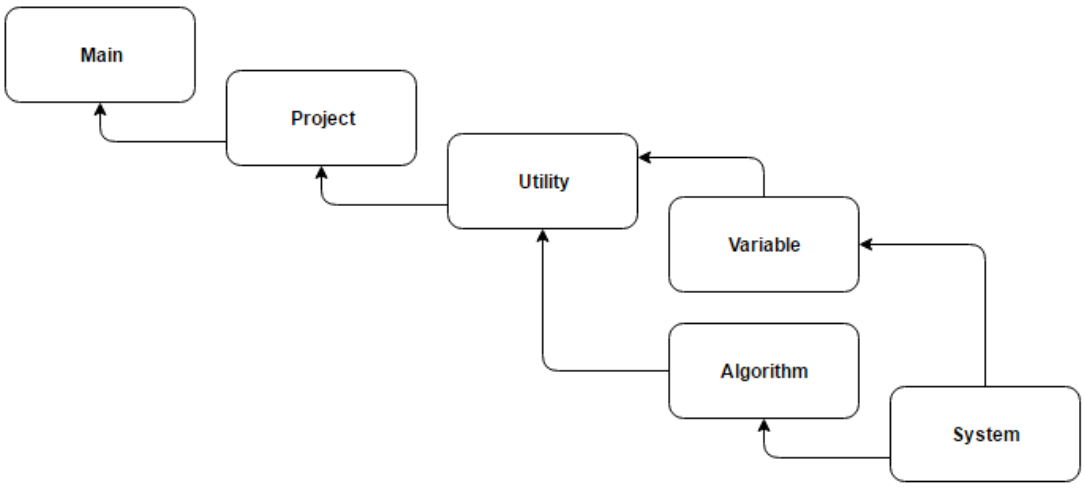


Figure 7.9: Structure of Embedded System Algorithm Group

The structure of the Embedded System Algorithm Group is presented in Fig.7.8. The embedded system codes are developed based on a set of open source code in microprocessor software developing toolkit. Each of the blocks presents a group of algorithm, the arrow indicates the referring relationship of the groups ("Project" is referred to by "Main", "Variable" is referred to by "Utility", etc.). The intention of creating such a structure is to provide the possibility that the algorithms can be clearly classified and modularized. The algorithm groups below the group level of "Project" are universal for all applications that uses the JRB board.

As the essential component of the control system, the algorithm group named "System" contains the open source code that provides the fundamental information of the microchip [40]. The "Algorithm" and "Variable" groups are independent to each other and include the basic algorithms (matrix calculations, attitude calculations, etc.) and definition of variables that are inherent the JRB board setup. The "Utility" group contains the commonly used hardware and software algorithms for the JRB board, which is referred by "Project", which contains the actual control algorithm and variable definition of Q-Baller. "Main" is the highest-level algorithm group that has access to all algorithms.

7.2.2. Utility Introductions and Initiations

Of the many utilities supported by STM32, the utilities we used in the Q-Baller control system only include:

1) PWM Signal Output (TIM1)
2) Quadrature Encoder Signal Input (TIM2, TIM3, TIM4, TIM5)
3) System Timer (TIM6, TIM7)

4) General Purpose Input/Output (GPIO)

5) Serial Data Transmission (USART1, USART2)

From above it can be realized that the three main categories of utility – Timer, GPIO and USART. GPIO is the default peripheral utility of microprocessor by reading in and giving out digital signals “0” and “1”. The Timers has multiple channel that can work alone or together to function as timer and counter. The USART is the simplest type of serial communication utility that receive and send data bits one at a time at a high frequency.

For our Q-Baller system, the PWM signal output function is supported by Timer 1. The PWM signals are used to control the power of the motors. The PWM signal output controls the voltage by turning the output switch between on and off at a very fast rate, which results in averaging the output voltage at the duty cycle percentage of the maximum output value as presented in Fig. 7.9 [41].

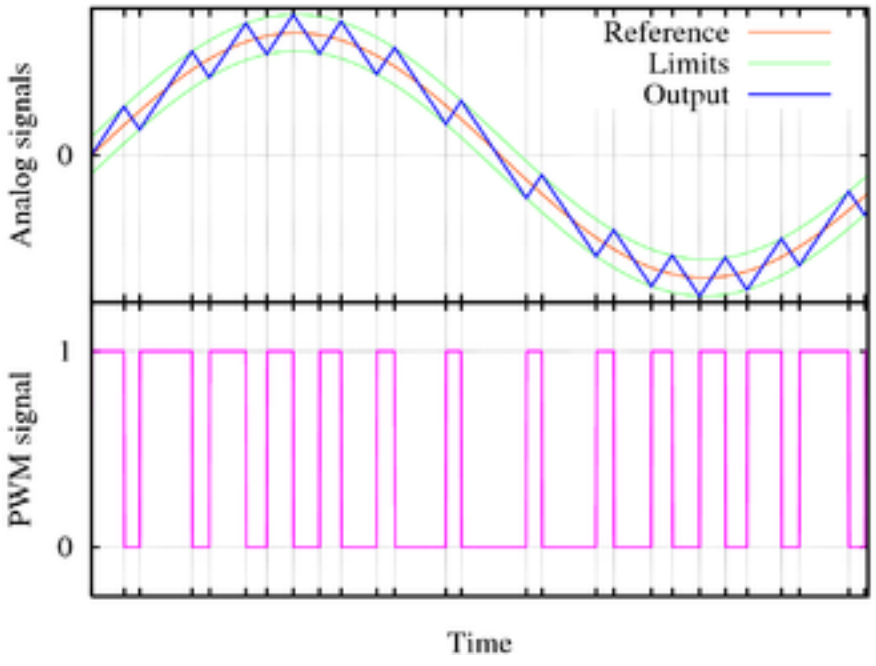


Figure 7.10: Concept of Pulse Width Modulation

The System Timer and Quadrature Encoder Input utilities operates in a similar way – both utilities use counters to count time segments or impulse numbers. Whenever the counters reach the designated cap number, the system will enter the interruption according to the overfill event and run the interrupting algorithm to either record the system time or the cycle of rotary encoders.

The USART [42] can receive and send data with other paired devices at a certain designed baud-rate. The baud-rate is the data transmission speed in bit/second. Both baud-rates selected for JY901 motion sensor and Bluetooth communication with control terminal are 115200 bit/s, which is one of the standard baud-rate value. When any of the devices receive a data byte during transmission, the device will enter the interruption function that will be designed to process the received data.

Finally, the GPIO ports are used to control the direction of motor based on the setup of the motor controller boards.

For most sensors and controllers, the standard signal voltage is 5V. However, the STM32 can only output signals at 3.3V. To solve this problem, the technique of Open Drain [43] is adopted. The designated pin of each output signal will be parallelly connected to the output port and the 5V source with exterior Pull-Up resistors. The pins will drain the current into the processor's inner circuit to output 0V or open circuiting the pins to output 5V. To do so, the pins must be able to handle the 5V load. The selection of Pull-Up resistors must also be moderate to prevent overflow of current. This features is designed into the JRB board to cater to the need of various applications.

The registers of STM32 for utility control must be configured to support the introduced functions. This part of the algorithms is called Utility Initialization which belongs to the "Utility" algorithm group. The initialization codes are specially designed for JRB board which can realize initialization conveniently and efficiently.

7.2.3. Algorithm Optimization

The control algorithm of Q-Baller is designed to realize the control strategy introduced in the previous chapters. However, since the microprocessor has limited storage to store all the algorithms and data, to prevent the squandering of algorithm and data storage space, the algorithm is designed according to following optimal principles:

1) Avoid usage of temporary variables and define the frequently used variables as global variables.

2) Modularize the algorithm by creating sub-functions that can be shared by multiple higher level functions.

3) Avoid long algorithms in interruption functions.

4) Manage the algorithm in correct order to avoid conflicts between different utilities.

The embedded system of Q-Baller is designed based on C language, which only support basic calculations of single values instead of matrixes. Since the control algorithm involves a lot of matrixes, an algorithm set of matrix mathematics is designed to efficiently calculate and manage matrix data.

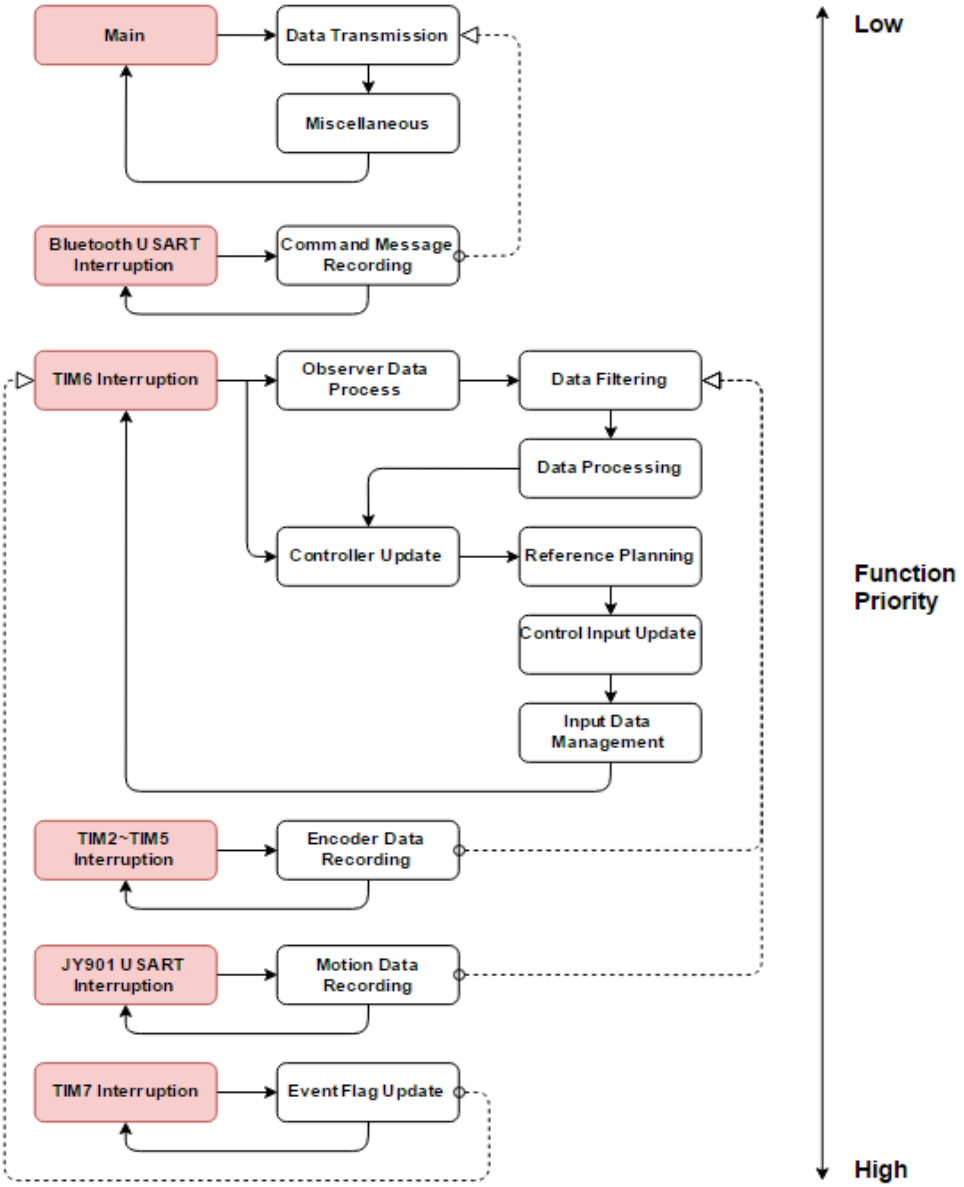


Figure 7.11: Embedded System Logic Structure

Complicate algorithms such as reference planning, data processing and controller updating will occupy a lot of calculation. However, the algorithm of data outgoing transmission through USART is different. The algorithm may occupy more time than other more complicate algorithms since its run time is determined by the designed transmission

speed. The algorithm will not end if the message has not been entirely sent out. To solve this problem, the two system timers are applied cooperatively to realize the logic structure in Fig. 7.10.

The priority of the main function and interruption functions are demonstrated. The low priority functions can only be interrupted by high priority functions. The solid arrow lines indicated the program running direction, and the dash arrow lines indicates the data usage direction between algorithm. The flow chart clearly explained that the system stability and control of Q-Baller is of a higher priority in the system.

The Timer 6 Interruption occurs at a designed frequency to perform the observer data process and the controller update. Any sensor event and interrupt the process and update the observer values. The Timer 7 has the highest interruption priority that is used to accurately record the system operating time.

The TIM7 Interruption can also generate event flags to indicate whether algorithms should be performed in TIM6 Interruption. For example, giving that he controller update at a frequency of 50 Hz, observer updates at a frequency of 100 Hz and the TIM6 Interruption occurs at 200 Hz, the event flag for observer update will only be activated every 4 interruption cycles, and the controller update will only be performed once per 4 interruption cycles. The frequencies of algorithms should be designed reasonably to fulfill the control requirements and avoid timing conflicts.

7.2.4. Operational Safety & Miscellaneous Algorithms

In addition to the major utility and control algorithms, the Q-Baller's safety has also been considered to prevent the prototype from damaging when any part of the system is

malfunctioning. The safety algorithm will start an alert that stop all control system outputs when the following situations happen:

1) The Q-Baller Body has tilted to extent that is beyond recovery.
2) The Q-Baller Body is operating at an abnormal velocity.
3) The feedback from the sensors has become inconsistent or erroneous.
4) The control calculation has become inconsistent and conflicting.

The other miscellaneous algorithms may include the recording of system operation data, state analysis or other algorithms that may consume a huge amount of time.

At the current stage, the prototype embedded system is complete. The controller can only realize fundamental utilities. The system still has many flaws to be amended and bugs to be fixes. Therefore, there is still a lot of spaces left for the embedded system to be improved. With the further development of Q-Ballers, more features will be added to the embedded system to make it more versatile and efficient.

7.3. Conclusion

In this chapter, the prototyping processes of Q-Baller and its embedded control system are introduced. A printed circuitry board of named “JMechW Robotic Board” has been designed to fulfill the prototype requirement of the electronic system. The materials, methods and techniques used in manufacturing the robot’s mechatronics system are explained, and the characteristics of the circuitry, the sensors and the microprocessor are elaborated in detail. The algorithms and logic structure of the embedded control system are also introduced. The prototype will be tested in practical experiments in the future.

CHAPTER 8. CONCLUSION & FUTURE PLANS

8.1. Conclusion of Current Work

The thesis discussed the research & development of Q-Baller. Technically speaking, the project cannot be considered complete without the physical experiments. However, the practical conditions for experiments has limited the project from going forward. At the current stage, when separately looking at each aspect of the research, we can conclude that the works done in design, modeling and control are successful.

The design of both mechanical and electronic system of Q-Baller has been proved effective. The Q-Baller's mechanical structure has been successfully prototyped based on the careful design, and the electronic system is proved the feasible since all actuators can be initiated and all feedback channels can be activated.

The dynamic modeling of Q-Baller can only be tested by the experiment, but the controller design based on the dynamic model has been proved successful. The CGS and EL-CGS has achieved great performance on the model. The reference planning algorithms has further improved the stability of the controller, especially for trajectory tracking. But since the project is about developing a product but not simply testing the theory, the true performance of the theoretic works can only be tested by the experiments.

Current preliminary experiments conducted on the prototype are limited to testing of the hardware. While the embedded system is working well and all individual components can be activated, the robot cannot function due to the selection of BDC Motors in the first stage of prototype experiment. The BDC Motors are selected as first try motors due to their

low price and relatively higher power output. The motors, however, presents high inertia and low control accuracy which both undermines the function of the robot. Therefore, no successful experimental data can be collected at the stage.

8.2. Plans for Future Works

The research project of Q-Baller has many potential and possibilities. The plans for future works is very hard to be determined specifically, but the general direction of the project development is presented below:

1) First and foremost, BDC motors will be replaced by BLDC motors. While BLDC motors are usually more expensive, they will provide reliable and accurate control performances. The dynamic model will be regenerated with the properties of the BLDC Motors, and thus new controllers can be designed for the updated Q-Baller prototype.
2) Preliminary experiments will then be carried out with the Q-Baller prototype. If the prototype has been proved feasible and successful, we will proceed to a few of more advanced experiments and collect the data to compared the difference between the theoretical and the actual dynamic models/controllers.

   The prototype will be troubleshot if failing the preliminary experiments. If proved improperly designed, the product will be redesigned and the current component selections may be reconsidered.
3) The current embedded system is developed based on process oriented programming. The system should be redesigned and improved with object-oriented programing to enhance its modularity and efficiency.

4) Advanced nonlinear control theory will be tested on the prototype if the previous plans are successfully carried out. Different controller will be applied to Q-Baller and their performance will be compared to further study the dynamic behavior of Q-Baller. Artificial intelligence technique such as artificial neural networks may also be applied for dynamic recognition and adaptive controller design.
5) More features are to be added to the Q-Baller. Robotic algorithms such as trajectory planning and motion designs will be added to the Q-Baller's programing. Additional sensors and actuators will be installed to make Q-Baller more versatile.

In conclusion, there are a lot of theories and possibilities that can be experimented with Q-Baller. The research of Ball-Bot can be very meaningful and challenging, and there may hardly be an end in exploiting the potential use of it.

So far, the research experience of Q-Baller has been a very meaningful and fruitful. We have learned a lot from this type of robot, especially in dynamics and control. The thesis may have come to an end here, but there are more to expect from the research & development of Q-Baller!

REFERENCES

[1]. T. B. Lauwers, G. A. Kantor, and R. L. Hollis: "A Dynamically Stable Single-Wheeled Mobile Robot with Inverse Mouse-Ball Drive", IEEE International Conference on Robotics and Automation. pp. 2884–2889, 2006.

[2]. M Kumagai, T Ochiai: "Development of a Robot Balancing on a Ball", International Conference on Control (Seoul, Korea: Automation and Systems): 433–438, 2008.

[3]. Simon Doessegger, Peter Fankhauser, Corsin Gwerder, Jonathan Huessy, Jerome Kaeser, Thomas Kammermann, Lukas Limacher, Michael Neunert: "Rezero, Focus Project Report". Autonomous Systems Lab, ETH Zurich, 2010.

[4]. Patent: "Omnidirectional wheel" by J Blumrich, Publication Number: US3789947A.

[5]. Patent: "Wheels for a course stable self-propelling vehicle movable in any desired direction on the ground or some other bases by Bengt Erland Ilon", Publication Number: US3876255A.

[6]. P. Muir and C. Neuman. "Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot." Robotics and Automation. Proceedings. 1987 IEEE International Conference on. Vol. 4. IEEE, 1987.

[7]. Hu, Jack, Zdzislaw Marciniak, and John Duncan, eds. "Mechanics of sheet metal forming". Butterworth-Heinemann, 2002.

[8]. John M. (Tim) Holt, Technical Ed; C. Y. Ho, Ed., "Structural Alloys Handbook, 1996 edition", CINDAS/Purdue University, West Lafayette, IN, 1996.

[9]. Yongxing Hao, Manxiang Miao, Xiaoyan Luo: “Mechatronic Power Transmission Control (Chinese Edition)”, ISBN-13: 978-560958620, Huazhong University of Science and Technology Press (2009)

[10]. A. Purna Chandra Rao, Y. P. Obulesh and Ch. Sai Babu: “Mathematical Modeling of BLDC Motor with Closed Loop Using PID Controller Under Various Loading Conditions”, ARPN Journal of Engineering and Applied Sciences, Vol. 7, No. 10, 2012.

[11]. A. Kapun, M. Curkovic, A. Hace, K. Jezernik: “Identifying dynamic model parameters of a BLDC motor”, Simulation Modelling Practice and Theory Vol. 16 P. 1254-1265, 2008.

[12]. Product Manual: “STM32F405XX & STM32F407XX Datasheet – Production data” by STMicroelectronics, DocID022152 Rev. 8, 2016.

[13]. Leutenegger, Stefan, and Péter Fankhauser. "Modeling and Control of a Ballbot", ETH E-Collection - ETH Zürich, 2010.

[14]. Donald T. Greenwood: “Principle of Dynamics (2nd Edition)”, ISBN-13: 978-0137099818, Pearson Education, 1987.

[15]. Ardakani, H. Alemi, and T. J. Bridges. "Review of the 3-2-1 euler angles: a yaw-pitch-roll sequence." Department of Mathematics, University of Surrey, Guildford GU2 7XH UK, Tech. Rep, 2010.

[16]. Phil Kim, “Rigid Body Dynamics for Beginners: Euler angles & Quaternions”, ISBN-13: 978-1493598205, CreateSpace Independent Publishing Platform, 2013.

[17]. Raffo, Guilherme V., Manuel G. Ortega, and Francisco R. Rubio. "An integral predictive/nonlinear H∞ control structure for a quadrotor helicopter." Automatica 46.1: 29-39, 2010.

[18]. Liao, Ching-Wen, et al. "Dynamic modeling and sliding-mode control of a ball robot with inverse mouse-ball drive." SICE Annual Conference, 2008. IEEE, 2008.

[19]. Ogata, Katsuhiko, and Yanjuan Yang. "Modern control engineering." (1970): 1.

[20]. Hermann, Robert, and Arthur Krener. "Nonlinear controllability and observability." IEEE Transactions on automatic control 22.5 (1977): 728-740.

[21]. Khalil, Hassan K. “Noninear Systems”. Prentice-Hall, New Jersey, 1996.

[22]. Luukkonen, Teppo. "Modelling and control of quadcopter." Independent research project in applied mathematics, Espoo (2011).

[23]. Skogestad, Sigurd. "Simple analytic rules for model reduction and PID controller tuning." Journal of process control 13.4 (2003): 291-309.

[24]. Naidu, D. Subbaram. “Optimal control systems”. CRC press, 2002.

[25]. Kwakernaak, Huibert, and Raphael Sivan. “Linear optimal control systems”. Vol. 1. New York: Wiley-interscience, 1972.

[26]. Boris Delaunay: “Sur la sphère vide”, Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles. 6: 793–800 (1934)

[27]. De Berg, Mark, et al. "Computational geometry." Computational geometry. Springer Berlin Heidelberg, 2000. 1-17.

[28]. Hille, Einar. Analytic function theory. Vol. 2. American Mathematical Soc., 2005.

[29]. Zhao, Zhen-Yu, Masayoshi Tomizuka, and Satoru Isaka. "Fuzzy gain scheduling of PID controllers." IEEE Transactions on Systems, Man, and Cybernetics 23.5 (1993): 1392-1398.

[30]. Johnson, Charles R. "A local Lyapunov theorem and the stability of sums." Linear Algebra and its Applications 13.1-2 (1976): 37-43.

[31]. Kendall, Maurice G. “A Course in the Geometry of n Dimensions”. Courier Corporation, 2004.

[32]. Higham, Desmond J., and Nicholas J. Higham. “MATLAB guide”. Society for Industrial and Applied Mathematics, 2005.

[33]. Zuo, Z. "Trajectory tracking control design with command-filtered compensation for a quadrotor." IET control theory & applications 4.11 (2010): 2343-2355.

[34]. Aguiar, A. Pedro, and Joao P. Hespanha. "Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty." IEEE Transactions on Automatic Control 52.8 (2007): 1362-1379.

[35]. Xin, Ming, Yunjun Xu, and Ricky Hopkins. "Trajectory control of miniature helicopters using a unified nonlinear optimal control technique." Journal of Dynamic Systems, Measurement, and Control 133.6 (2011): 061001.

[36]. Brandrup, Johannes, et al., eds. Polymer handbook. Vol. 7. New York etc: Wiley, 1989.

[37]. Khandpur, Raghbir Singh. Printed circuit boards: design, fabrication, assembly and testing. Tata McGraw-Hill Education, 2005.

[38]. Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." Journal of basic Engineering 82.1 (1960): 35-45.

[39]. OpenEdv Team: “STM32F4 Development Manual (Chinese Version)”, 2015.

[40]. MCD Application Team: “CMSIS Cortex-M4 Device Peripheral Access Source Code”, STMicroelectronics, 2014.

[41]. Barr, Michael. "Pulse width modulation." Embedded Systems Programming 14.10 (2001): 103-104.

[42]. Gulick, Dale E., Terry G. Lawell, and Charles Crowe. "Enhanced universal asynchronous receiver-transmitter." U.S. Patent No. 4,949,333. 14 Aug. 1990.

[43]. Chengson, David P., and Robert A. Conrad. "Multi-configurable push-pull/open-drain driver circuit." U.S. Patent No. 5,811,997. 22 Sep. 1998.

APPENDIX I. DYNAMIC MODELING CODE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Dynamic Modeling of Q-Baller in MATLAB
%By Jiamin Wang
%Latest Update: 2017/4/14
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;%Start Code Runtime Timer
digits(32);%Set Calculation Precision as Single Precision Float

syms    Hx Hy Hz g M m R r T_0 n_0 U_0...
       J_w b_w b_BL b_bL...
       j_XX j_XY j_XZ j_YX j_YY j_YZ j_ZX j_ZY j_ZZ...
       j_xx j_xy j_xz j_yx j_yy j_yz j_zx j_zy j_zz...
       b_XX b_XY b_XZ b_YX b_YY b_YZ b_ZX b_ZY b_ZZ...
       b_xx b_xy b_xz b_yx b_yy b_yz b_zx b_zy b_zz...%System Constants

syms    T_x1y1 T_x1y0 T_x0y0 T_x0y1 U_x1y1 U_x1y0 U_x0y0 U_x0y1...
       w_x1y1 w_x1y0 w_x0y0 w_x0y1 w1_x1y1 w1_x1y0 w1_x0y0 w1_x0y1...
       A0 B0 C0 a0 b0 c0 X0 Y0 Z0 x0 y0 z0 ...
       A1 B1 C1 a1 b1 c1 X1 Y1 Z1 x1 y1 z1 ...
       A2 B2 C2 a2 b2 c2 X2 Y2 Z2 x2 y2 z2 ...%System Variables

syms    T T_m F_w T_w F_GX F_GY N_G N0 F_dx F_dy F_dz T_dx...
       T_dy T_dz F_DX F_DY F_DZ T_DX T_DY T_DZ; %Intermediates

%Disturbance Forces and Torques
F_d=[F_dx;F_dy;F_dz];%Disturbance Force on the Ball
T_d=[T_dx;T_dy;T_dz];%Disturbance Torque on the Ball
F_D=[F_DX;F_DY;F_DZ];%Disturbance Force on the Body
T_D=[T_DX;T_DY;T_DZ];%Disturbance Force on the Body
GM=[0 0 -g*M];      %Gravity Force

%Variable States
Q0=[A0;B0;C0;x0;y0;c0];
Q1=[A1;B1;C1;x1;y1;c1];
Q2=[A2;B2;C2;x2;y2;c2];
Q0T=Q0.';
```

```
Q1T=Q1.';
Q2T=Q2.';

CLGA=[1 0 0; 0 cos(A0) -sin(A0); 0 sin(A0) cos(A0)];%X Rolling
CLGB=[cos(B0) 0 sin(B0); 0 1 0; -sin(B0) 0 cos(B0)];%Y Pitching
CLGC=[cos(C0) -sin(C0) 0; sin(C0) cos(C0) 0; 0 0 1];%Z Yawing

CGLA=CLGA.'; %Transpose of CLGA
CGLB=CLGB.'; %Transpose of CLGB
CGLC=CLGC.'; %Transpose of CLGC

C_L_to_G=CLGC*CLGB*CLGA;%Conversion Matrix from Frame L to G
C_G_to_L=CGLA*CGLB*CGLC;%Conversion Matrix from Frame G to L

C_O_to_L=subs(C_G_to_L,C0,0);%Conversion Matrix from Frame O to L
C_L_to_O=subs(C_L_to_G,C0,0);%Conversion Matrix from Frame O to L

C_O_to_G=CLGC;%Conversion Matrix from Frame O to G
C_G_to_O=CGLC;%Conversion Matrix from Frame G to O

Mj=[1 0 -sin(B0);...
    0 cos(A0) sin(A0)*cos(B0);...
    0 -sin(A0) cos(B0)*cos(A0)];%Jacobian Matrix
MjT=Mj.';                   %Transpose of Mj

IX=sqrt(6)/4;IY=sqrt(6)/4;IZ=1/2;IA=sqrt(2)/2;IB=sqrt(2)/4;IC=sqrt(3)/2;
%Intermediate Constants Related to the Geometry of the Robot

M_M=[IX -IY -IZ;...
    -IX -IY IZ;...
    -IX IY -IZ;...
    IX IY IZ]';%Motor Torque Postive Direction
M_R=[IB -IB IC;...
    IB IB IC;...
    -IB IB IC;...
    -IB -IB IC]';%Motor Contact Point Position Direction
M_L=[IA IA 0;...
    IA -IA 0;...
    -IA -IA 0;...
    -IA IA 0]'; %Motor Contact Point Velocity Direction
T_M=[T_x1y1;...
```

```
    T_x1y0;...
    T_x0y0;...
    T_x0y1]; %Torque from Motor X+Y+;X+Y-;X-Y-;X-Y+ respectively
T_W=-(R/r)*M_M*T_M; %Sume of Torque ccting on the Ball

J_B=[j_XX j_XY j_XZ; j_YX j_YY j_YZ; j_ZX j_ZY j_ZZ];%Rotary Inertia of
Body
J_b=[j_xx j_xy j_xz; j_yx j_yy j_yz; j_zx j_zy j_zz];%Rotary Inertia of
Ball

B_BR=[b_XX b_XY b_XZ; b_YX b_YY b_YZ; b_ZX b_ZY b_ZZ];%Rotary Damper of
Body
B_bR=[b_xx b_xy b_xz; b_yx b_yy b_yz; b_zx b_zy b_zz];%Rotary Damper of
Ball

B_BL=b_BL;%Translational Damper on Body
B_bL=b_bL;%Translational Damper on Ball

VecR=R*M_R;%Vectors from the Center of Ball to the Wheel Contact Points
VecG=[0;0;-R];%Vector from the Center of Ball to the Ground
VecH=[Hx;Hy;Hz];%Vector from the Center of Ball to the Center of Mass of
Body

%Velocities
VO=[x1;y1;0];%Velocity in Orientation Frame
VOT=[x1 y1 0];
VG=C_O_to_G*[x1;y1;0];%Velocity in Ground Frame
VGT=[x1 y1 0]*C_G_to_O;

WBL=Mj*Q1(1:3);%Angular Velocity of Robot Body
WBLT=Q1T(1:3)*MjT;
WbO=[-Q1(5)/R;Q1(4)/R;Q1(6)];%Angular Velocity of Ball
WbOT=[-Q1(5)/R Q1(4)/R Q1(6)];
WbL=C_O_to_L*WbO;%Angular Velocity of Ball in Local Frame
WbLT=WbOT*C_L_to_O;
WbG=C_O_to_G*WbO;%Angular Velocity of Ball in Ground Frame
WbGT=WbOT*C_G_to_O;

%Kinematic Energy
TL=0.5*(VGT*(M+m)*VG);%Translational Kinematic Energy
TR=0.5*(WBLT*J_B*WBL)+0.5*(WbOT*J_b*WbO);%Rotary Kinematic Energy
```

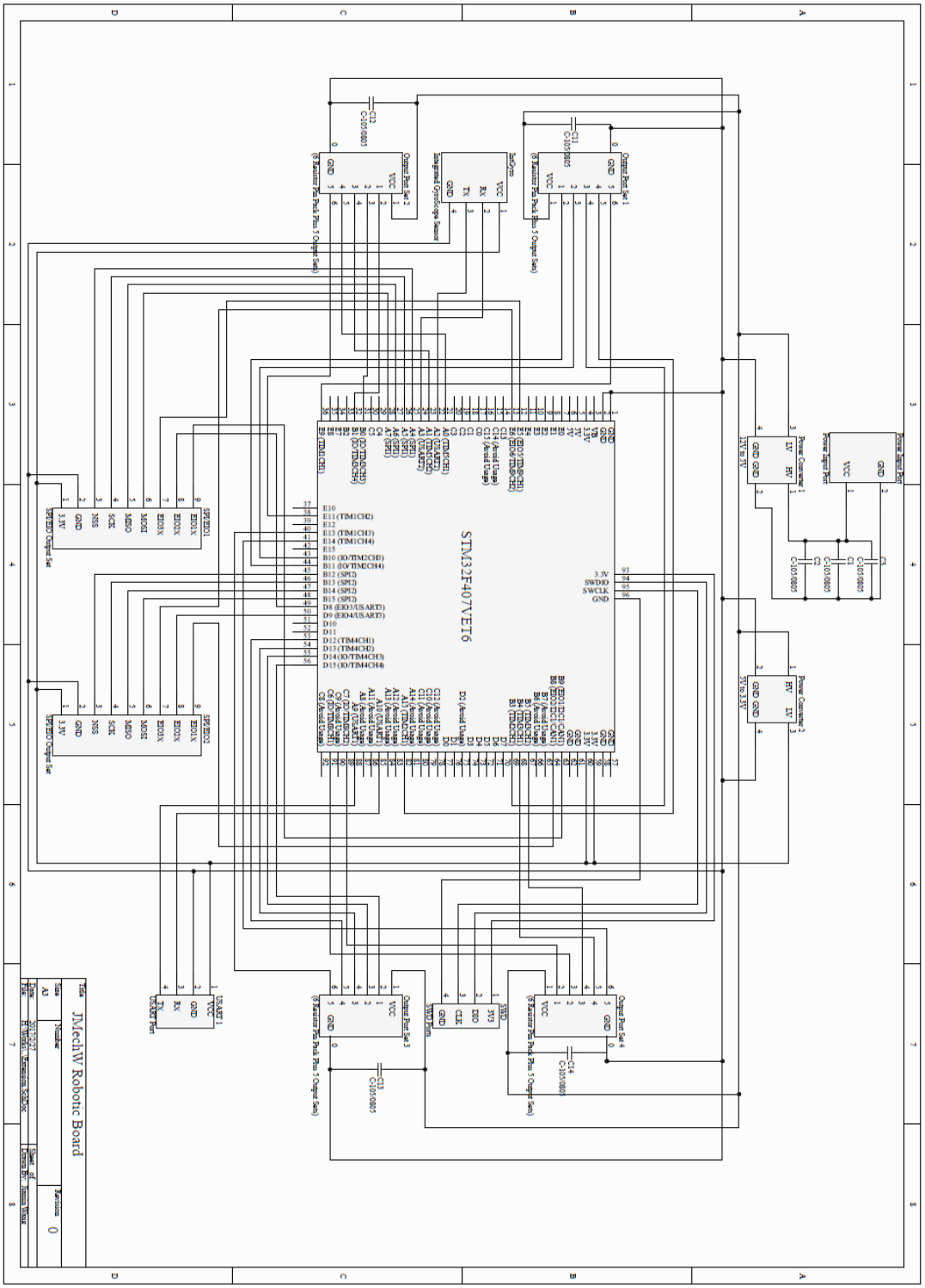
```
TC=M*(VGT*C_L_to_G)*(cross(WBL,VecH));%Coupling Kinematic Energy Term

%Potential Energy
V=-GM*C_L_to_G*VecH;%Potential Energy

%Lagrangian Factor
L=TL+TR+TC-V;%Total Mechanic Energy

%Energy Dissipation
DR=0.5*VOT*(C_L_to_O*B_BL+B_bL)*VO+0.5*WBLT*B_BR*WBL+0.5*WbOT*B_bR*WbO;

%Sum of Input (Control Input & Noise)
Qdt=WbOT*C_L_to_O*(T_W+C_G_to_L*T_d)+WBLT*(-
T_W+C_G_to_L*T_D)+VGT*(F_D+F_d)-DR;

%Lagrangian Equation
EQLdq_dt=jacobian(jacobian(L,Q1),Q1)*Q2+jacobian(jacobian(L,Q1),Q0)*Q1;
EQLdqT=jacobian(L,Q0);
EQLdq=[EQLdqT(1);EQLdqT(2);EQLdqT(3);EQLdqT(4);EQLdqT(5);EQLdqT(6)];
EQL=EQLdq_dt-EQLdq;%Original Left Hand Side Term
EQRT=jacobian(Qdt,Q1);
EQR=EQRT.';%Original Left Hand Side Term

%Arranged Lagrangian Equations
EQL=EQL-EQR+jacobian(EQR,[T_d;T_D;F_D;F_d])*[T_d;T_D;F_D;F_d];
EQR=EQR-EQR+jacobian(EQR,[T_d;T_D;F_D;F_d])*[T_d;T_D;F_D;F_d];

%Motor Velocities
%Acheived from w_x1y1*r-
M_L(:,1)'*cross(WBL,VecR(:,1))=M_L(:,1)'*cross(C_O_to_L*WbO,VecR(:,1))
w_x1y1=M_L(:,1)'*(cross(C_O_to_L*WbO,VecR(:,1))-cross(WBL,VecR(:,1)))/r;
w_x1y0=M_L(:,2)'*(cross(C_O_to_L*WbO,VecR(:,2))-cross(WBL,VecR(:,2)))/r;
w_x0y0=M_L(:,3)'*(cross(C_O_to_L*WbO,VecR(:,3))-cross(WBL,VecR(:,3)))/r;
w_x0y1=M_L(:,4)'*(cross(C_O_to_L*WbO,VecR(:,4))-cross(WBL,VecR(:,4)))/r;

%Motor Accelerations
w1_x1y1=jacobian(w_x1y1,Q0)*Q1+jacobian(w_x1y1,Q1)*Q2;
w1_x1y0=jacobian(w_x1y0,Q0)*Q1+jacobian(w_x1y0,Q1)*Q2;
w1_x0y0=jacobian(w_x0y0,Q0)*Q1+jacobian(w_x0y0,Q1)*Q2;
w1_x0y1=jacobian(w_x0y1,Q0)*Q1+jacobian(w_x0y1,Q1)*Q2;
```

```
%Modeling of Motors
w=[w_x1y1;w_x1y0;w_x0y0;w_x0y1];
w1=[w1_x1y1;w1_x1y0;w1_x0y0;w1_x0y1];
U=[U_x1y1;U_x1y0;U_x0y0;U_x0y1];
T_MotorModel=(T_0/U_0)*U-J_w*w1-b_w*w-(60*T_0/(2*pi*n_0))*w;

%Complete System
EQR=simplify(EQR);
EQL=simplify(subs(EQL,T_M,T_MotorModel));

%Naming String for Model Function Compilation
Name={'StandardModel';'RandomModel';...
    'StandardModelSSMatrix';'RandomModelSSMatrix'};

%Generation of Standard Model and Random Model
for ii=0:1 %ii=0 for Standard Model and ii=1 for Random Model

    %System Constant Symbols
    CONS=[Hx Hy Hz g M m R r T_0 n_0 U_0...
          J_w b_w b_BL b_bL...
          j_XX j_XY j_XZ j_YX j_YY j_YZ j_ZX j_ZY j_ZZ...
          j_xx j_xy j_xz j_yx j_yy j_yz j_zx j_zy j_zz...
          b_XX b_XY b_XZ b_YX b_YY b_YZ b_ZX b_ZY b_ZZ...
          b_xx b_xy b_xz b_yx b_yy b_yz b_zx b_zy b_zz];

    %System Constant Properties
    VALS=vpa([0 0 0.107 9.81 6.4 1.5 0.1 0.024 2.5 2000 12 ...
          1.25*10^(-5) 0.005 0.008 0.002 ...
          0.1488 0 0 0 0.1512 0 0 0 0.0746...
          0.00975 0 0 0 0.00975 0 0 0 0.00975...
          0.02 0 0 0 0.02 0 0 0 0.01...
          0.01 0 0 0 0.01 0 0 0 0.01]+...%Properties of Standard Model
          [0.1 0.1 0.1 0 6.4 1.5 0.1 0.024 2.5 2000 0 ...
          1.25*10^(-5) 0.005 0.008 0.002 ...
          0.1488 0 0 0 0.1512 0 0 0 0.0746...
          0.00975 0 0 0 0.00975 0 0 0 0.00975...
          0.02 0 0 0 0.02 0 0 0 0.01...
          0.01 0 0 0 0.01 0 0 0 0.01]*...
          diag([random('unif',-0.3,-0.2,[2,1]);...
          random('unif',0.2,0.3,[1,1]);...
          random('unif',-0.3,0.3,[5,1]);...
```

```
      random('unif',-0.3,0,[2,1]);...
      random('unif',-0.3,0.3,[41,1])])*ii,4);%Random Property Scaler

%Digit Compression to Avoid Numerical Error in Calculation
%Due to the Complexity of Model
digits(4);
VALS=double(VALS)*1e6;
digits(16);
VALS=double(VALS);

%Save Standard and Random Property Datas
save(char(strcat(pwd,'\',Name(ii+1),'Data.mat')),'VALS');

%Replace Symbols With Property Values
EQLreal=simplify(subs(EQL,CONS,sym(VALS,'r')/1e6));
EQRreal=simplify(subs(EQR,CONS,sym(VALS,'r')/1e6));

%Standard Form
Matrix_M=simplify(jacobian(EQLreal,Q2));
invMatrix_M=simplify(inv(Matrix_M));

%Simplication of Symbolic Equations
%Has to be Finished Seperately to Prevent Error Due to Complexity
SEQreal=invMatrix_M*simplify(EQRreal-EQLreal+Matrix_M*Q2);
SEQreal1=simplify(SEQreal(1));
SEQreal2=simplify(SEQreal(2));
SEQreal3=simplify(SEQreal(3));
SEQreal4=simplify(SEQreal(4));
SEQreal5=simplify(SEQreal(5));
SEQreal6=simplify(SEQreal(6));

%Full Model
SEQreal=[SEQreal1;SEQreal2;SEQreal3;SEQreal4;SEQreal5;SEQreal6];

%Ideal Model (With No Disturbances)
SEQideal=subs(SEQreal,[T_d;T_D;F_D;F_d],zeros(12,1));

%State Space Matrixes A and B
SSMA=[zeros(6,6) eye(6,6);jacobian(SEQideal,[Q0;Q1])];
SSMB=[zeros(6,4);jacobian(SEQideal,U)];
```

```
    %Save the Data of Different Models
    save(char(strcat(pwd,'\',Name(ii+1),'Real.mat')),'SEQreal');
    save(char(strcat(pwd,'\',Name(ii+1),'Ideal.mat')),'SEQideal');
    save(char(strcat(pwd,'\',Name(ii+3),'A.mat')),'SSMA');
    save(char(strcat(pwd,'\',Name(ii+3),'B.mat')),'SSMB');

    %Generate the Symbolic Matlab Function
    FullODE=[Q1;SEQreal];

matlabFunction(FullODE,'file',char(strcat(Name(ii+1),'Fcn')),'vars',{[Q0
;Q1],[U],[F_D],[F_d],[T_D],[T_d]});

end
toc;%Record Code Runtime

%Generate the Precompiled .mex Files for Simulation Efficiency
CoderType={coder.typeof(0,[12,1]),...
coder.typeof(0,[4,1]),...
coder.typeof(0,[3,1]),...
coder.typeof(0,[3,1]),...
coder.typeof(0,[3,1]),...
coder.typeof(0,[3,1])};
pathstandard=char(strcat(pwd,'\',Name(1),'Fcn.m'));
pathrandom=char(strcat(pwd,'\',Name(2),'Fcn.m'));
codegen -config:mex StandardModelFcn -args CoderType
codegen -config:mex RandomModelFcn -args CoderType
```

APPENDIX II. JMECHW ROBOTIC BOARD SCHEMETICS



APPENDIX III. IO SETUP FOR JRB DESIGN

Pin Designator	Name of Port	STM32 (JRB) Utility Description
1	GND	
2	GND	
3	VB	
4	3.3V	
5	5V	
6	5V	
7	E0	
8	E1	
9	E2	
10	E3	
11	E4	
12	E5	AF3 (EIO5/TIM9CH1)
13	E6	AF3 (EIO6/TIM9CH2)
14	C13	
15	C14	OSC **(Avoid Usage)**
16	C15	OSC **(Avoid Usage)**
17	C0	
18	C1	
19	C2	
20	C3	
21	A0	AF2 (TIM5CH1)
22	A2	AF7 (USART2)
23	A1	AF2 (TIM5CH2)
24	A3	AF7 (USART2)
25	A4	AF5 (SPI1)
26	A5	AF5 (SPI1)
27	A6	AF5 (SPI1)
28	A7	AF5 (SPI1)
29	C4	
30	C5	
31	B0	AF2 (IO/TIM3CH3)
32	B1	AF2 (IO/TIM3CH4)
33	B2	
34	E7	

Pin Designator	Name of Port	STM32 Utility Description
35	E8	
36	E9	AF1 (TIM1CH1)
37	E10	
38	E11	AF1 (TIM1CH2)
39	E12	
40	E13	AF1 (TIM1CH3)
41	E14	AF1 (TIM1CH4)
42	E15	
43	B10	AF1 (IO/TIM2CH3)
44	B11	AF1 (IO/TIM2CH4)
45	B12	AF5 (SPI2)
46	B13	AF5 (SPI2)
47	B14	AF5 (SPI2)
48	B15	AF5 (SPI2)
49	D8	AF7 (EIO3/USART3)
50	D9	AF7 (EIO4/USART3)
51	D10	
52	D11	
53	D12	AF2 (TIM4CH1)
54	D13	AF2 (TIM4CH2)
55	D14	AF2 (IO/TIM4CH3)
56	D15	AF2 (IO/TIM4CH4)
57	GND	
58	GND	
59	3.3V	
60	3.3V	
61	GND	
62	GND	
63	B9	AF4/AF9 (EIO1/I2C1/CAN1)
64	B8	AF4/AF9 (EIO2/I2C1/CAN1)
65	B7	EEPROM **(Avoid Usage)**
66	B6	EEPROM **(Avoid Usage)**
67	B5	AF2 (TIM3CH2)
68	B4	AF2 (TIM3CH1)
69	B3	AF1 (TIM2CH2)
70	D7	
71	D6	
72	D5	

Pin Designator	Name of Port	STM32 Utility Description
73	D4	
74	D3	
75	D2	SD Card **(Avoid Usage)**
76	D1	
77	D0	
78	C12	SD Card **(Avoid Usage)**
79	C10	SD Card **(Avoid Usage)**
80	C11	SD Card **(Avoid Usage)**
81	A14	SWD **(Avoid Usage)**
82	A15	AF1 (TIM2CH1)
83	A12	USB **(Avoid Usage)**
84	A13	SWD **(Avoid Usage)**
85	A10	AF7 (USART1)
86	A11	USB **(Avoid Usage)**
87	A8	SD Card **(Avoid Usage)**
88	A9	AF7 (USART1)
89	C7	AF3 (IO/TIM8CH2)
90	C9	SD Card **(Avoid Usage)**
91	C6	AF3 (IO/TIM8CH1)
92	C8	SD Card **(Avoid Usage)**
93	3.3V	Jtag Line
94	SWDIO	Jtag Line
95	SWCLK	Jtag Line
96	GND	Jtag Line

Note: The JRB is designed based on the STM32 Microchip System Board by *Vcc-Gnd Electronics* (http://www.vcc-gnd.com/).

ENDING REMARK

For more resources, documentations and future updates about Q-Baller, please visit:

https://github.com/JMechW/Q-Baller-Ballbot-Project/

For more about the Q-Baller Project and research cooperation opportunity, please contact us through the information left on the above address.

Thank you!

Jiamin Wang