Wrocław University of Science and Technology
Faculty of Electronics, Photonics and Microsystems

FIELD OF STUDY: Control Engineering and Robotics (AIR)

MASTER THESIS

TITLE OF THESIS:
Control System for
Two HOG Wheel Mobile Robot

AUTHOR:
Eryk Możdżeń

SUPERVISOR:
Robert Muszyński PhD,
Department of Cybernetics and Robotics

WROCŁAW 2025

Contents

For typesetting this thesis, the LaTeX document preparation system has been used. LaTeX has been developed by L. Lamport [Lam94], and is an overlay on top of the TeX system [Knu86a, Knu86b]. Mathematical fonts called AMS Euler which have been used in this document, have been commissioned by the American Mathematical Society and designed by H. Zapf [KZ86] with the assistance of D. Knuth and his students. The text font, called URW Palladio, is a clone of the Zapf font family named Palatino [LPn05], and in the author's opinion, it harmonizes exceptionally well with the design of the AMS Euler font. The authors of the thesis template are R. Muszyński and R. Orozco [MO22], who utilized the mgr.cls document class created by A. Ratajczak [Rat08].

[Knu86a] D. E. Knuth, The TeX book, volume A of Computers and Typesetting. Addison-Wesley, Reading, 1986.

[Knu86b] D. E. Knuth, TeX: The Program, volume B of Computers and Typesetting. Addison-Wesley, Reading, 1986.

[KZ86] D. E. Knuth, H. Zapf, AMS Euler — A new typeface for mathematics. Scholary Publishing, 20:131–157, 1986.

[Lam94] L. Lamport, LaTeX: A Document Preparation System. Addison--Wesley, Reading, 1994.

[LPn05] Linotype Palatino nova: A classical typeface redesigned by Hermann Zapf, Linotype Library GmbH, 2005.

[MO22] R. Muszyński, R. Orozco, Examples and formatting guidelines for a thesis. https://kcir.pwr.edu.pl/~mucha/Pracki/praca_dyplomowa_wzor.pdf, 2022.

[Rat08] A. Ratajczak, Document class – mgr.cls. http://diablo.kcir.pwr.edu.pl/~ar/LaTeX/mgr.php, 2008.

Chapter 1

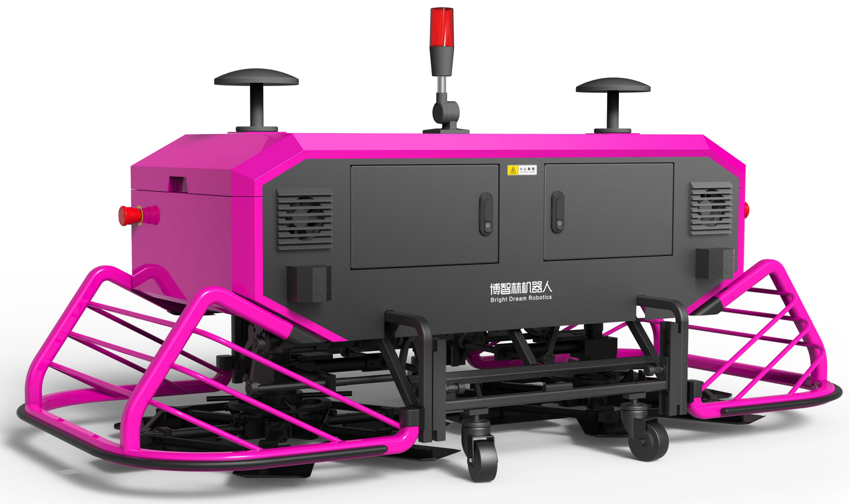
Introduction

Nowadays, mobile robotic systems are widely used to manage warehouses and to support production lines in factories around the world. Robots often operate in well-known environments specifically prepared for their operation, e.g., mobile robots on flat concrete floors in warehouses. In order to perform agile movements on flat ground, it is common to use omnidirectional wheels as the main drive system for UGV platforms [KSBT$^+$17].

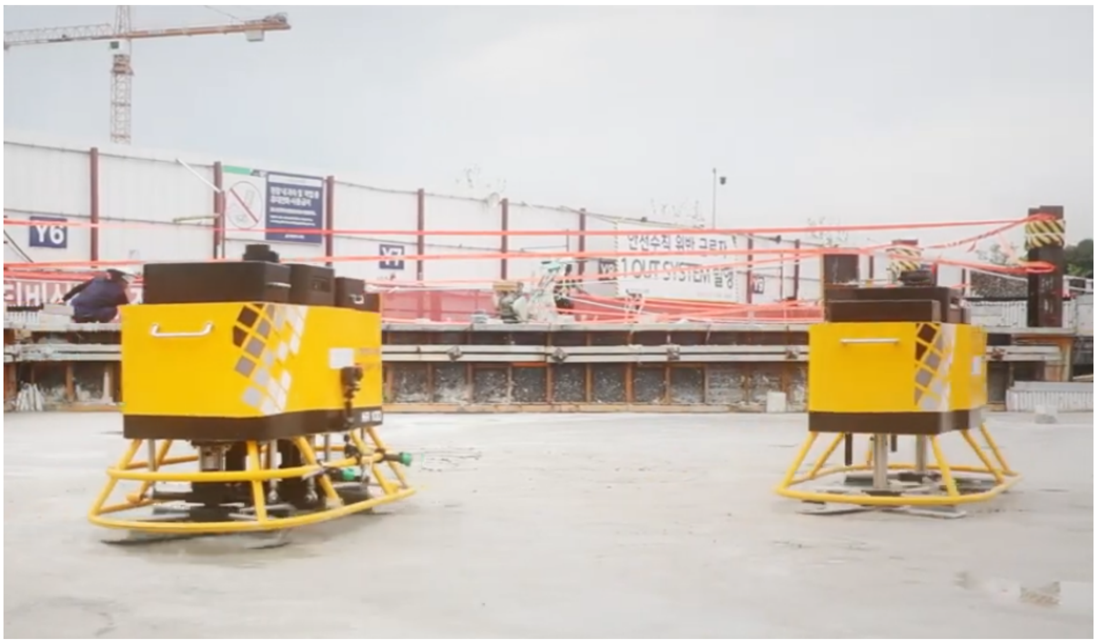
The Hemispherical Omnidirectional Gimballed (HOG) wheel from 1938 [mec38], called also "Singularity Drive System", has been a topic of research in robotics applications in the past few years [Ack, Ryb13, Jon14, Jon17, Boc19]. In theory, the use of constantly rotating hemispheres, each with a variable orientation, enables motion with high acceleration in any chosen direction. Additionally, massive hemispheres with high inertia can act as a kinetic energy bank for the platform, allowing the use of less powerful motors without compromising the robot's agility.

A similar type of propulsion can be observed in power troweling machines, which are widely used in the finishing of concrete surfaces. This branch of the industry has already been robotized, and examples of such robots are presented in Figure 1.1. In contrast, the HOG drive relies on a single point of contact with the ground, which has the advantage of allowing a relatively simple mathematical description of the system. Nevertheless, sufficiently large hemispheres should also operate reliably in muddy and sandy environments, similarly to trowelling machines.

The goal of this thesis is to modernize the existing mechanical design of a two HOG wheel mobile robot, develop a hardware platform that enables the implementation of advanced control algorithms for such a robot, and conduct a qualitative analysis of the performance of selected control algorithms that had previously been tested in simulation environments only. The content of this work is organized as follows. Chapter 2 contains a description of the robot, including mechanical modifications, and custom controller module. The description of the tested algorithms are presented in Chapter 3. Chapter 4 presents the results of the conducted experiments along with detailed conclusions. Chapter 5 contains a summary and the key conclusions drawn from working with the HOG drive technology.



(a) DERUTU robot (2024) [DER, ifd]



(b) ROBOBLOCK robots (2022) [ROBa]

Figure 1.1: Autonomous concrete troweling robots

Chapter 2

Platform Description

Robots with two HOG wheels are not widely available. For the purposes of this work, the *Hogger*^{2} robot (shown in Figure 2.1) originally designed and elaborated as a part of the thesis [Gór17] was expanded and redesigned. This chapter includes the description of the dedicated electronic controller module, the developed software, the mechanical modifications and solutions of other issues identified during the development.

A HOG wheel, shown in Figure 2.2, it a type of drive system that allows movement in any direction using constantly spinning hemisphere, or part of it. In perfect conditions, hemisphere touches the ground in exactly one contact point P. By tilting the axis of rotation from the vertical position by small amount α, one can directly control the point of contact with the ground, effectively creating a wheel with variable radius r, angular velocity ω, and heading (not shown in Figure).

In the *Hogger*^{2} robot two HOG wheels are applied with an additional one passive Caster wheel that acts as a third support point. Assuming that the robot moves on a plane, it's actual pose can be described using the coordinates x and y of the selected HOG wheel contact point and the orientation of the robot body. To obtain a complete robotic platform, it was necessary to equip the robot with perception and control systems.

2.1 Requirements, Decisions and Assumptions

Integrated controller circuit acts as central decision unit mounted on the robot body. First version of the hardware design was developed by the author as part of the "Intermediate Project" course and described in reports [Moż24, Moż25]. Electronics circuits design for the *Hogger*^{2} robot is shown in Figure 2.3. The design consist of main microcontroller control unit, actuators for two HOG wheels, sensors needed for environment perception, and additional microcontroller with wireless communication capabilities. The overall design reduced the number of wires between components to a useful minimum. Electronics schematic as well as PCB design of the controller module is attached in Appendix A.

From the control task point of view the robot consists of two BLDC motors, for spinning the hemispheres, and four servomotors, for tilting them. For the sake of simplicity and robustness, it was decided to incorporate the BLDC motor controllers as a part of

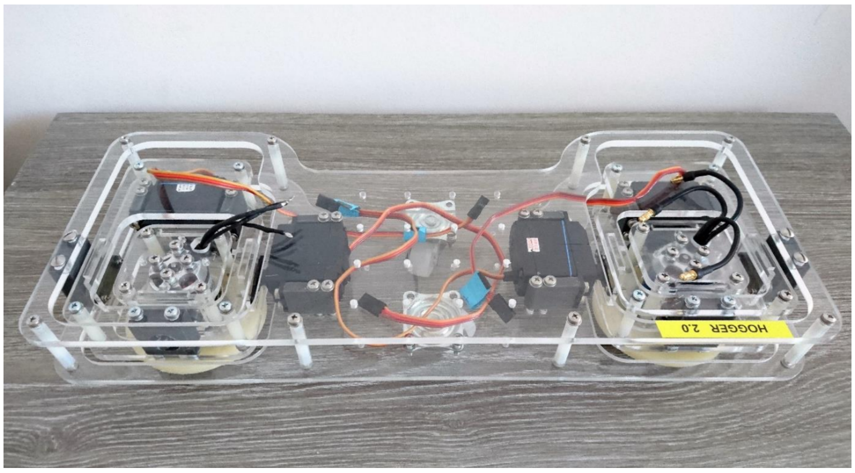


Figure 2.1: The original *Hogger*^{2} robot [Gór17]

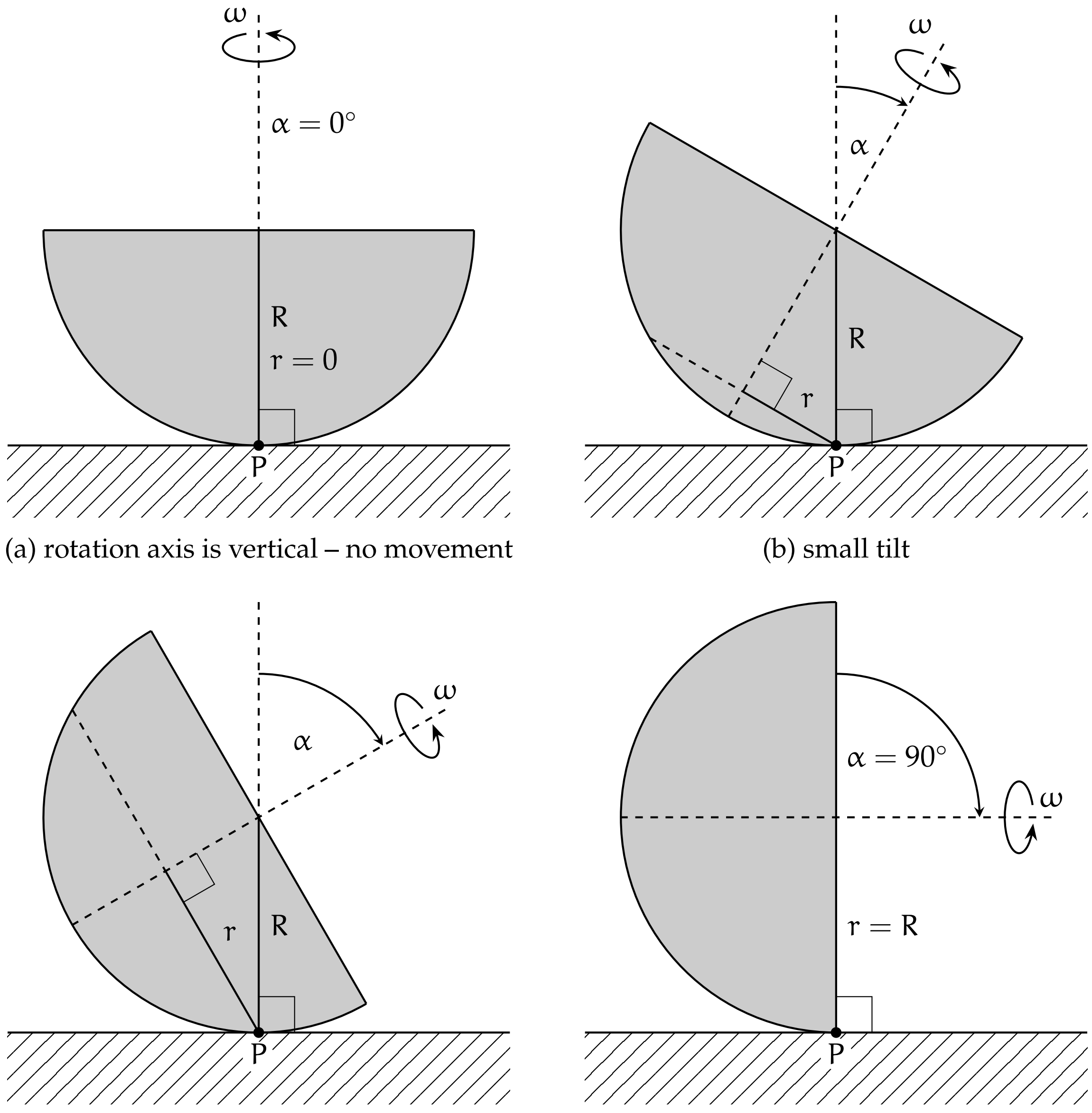
the controller module, rather than relying on off-the-shelf solutions. It was assumed that hemispheres should rotate at relatively high speeds, so a "sensorless" method for driving BLDC motors was selected [LCA]. This approach enables measurement of the motor's actual velocity without the need to install additional encoders on the motor shaft.

In order to perform advanced trajectory tracking algorithms, information about current position and velocity of the robot and its joints must be available to the robot embedded microcontroller in real time. It was decided that the state vector would be estimated using sensor fusion techniques, without correction from external systems. A further reduction in wiring was achieved by placing most of the sensors inside the controller module itself, creating an all-in-one solution. Both the solutions are described below.

2.2 Mechanical Modifications

In order to conduct experiments with the existing robotic platform, its components had to be supplemented and modified. Mechanical components added to the robot are shown in Figure 2.4. The optical sensor was mounted into conical part, shown in Figure 2.5, to maximize height above ground without limiting the field of view. The caster wheel, the third support point, was shifted away from the main body of the robot to achieve a stable construction. The 3S LiPo 2000 mAh battery was placed on the center wheel at the back to act as a counterweight to the optical sensor cone located at the front. The controller module casing, shown in Figure 2.6, was mounted on top of the robot using mainly existing mounting points.

The servo drives originally used in the robot were not capable of precise control of the gimbal axes. The PWM interface was not capable of sending telemetry containing the actual position and velocity values of the drives. As a replacement, four Dynamixel AX-12A [ROBb] servomotors were mounted to drive all gimbal axes. In addition, the existing damaged BLDC motors were replaced with KAVAN C2830-750 motors [KAV]. The connection between the motor shaft and the hemispherical wheel, shown in Fig-



(a) rotation axis is vertical – no movement

(b) small tilt

(c) bigger tilt

(d) maximum tilt – standard wheel

Figure 2.2: A HOG wheel tilting principle (cross section)

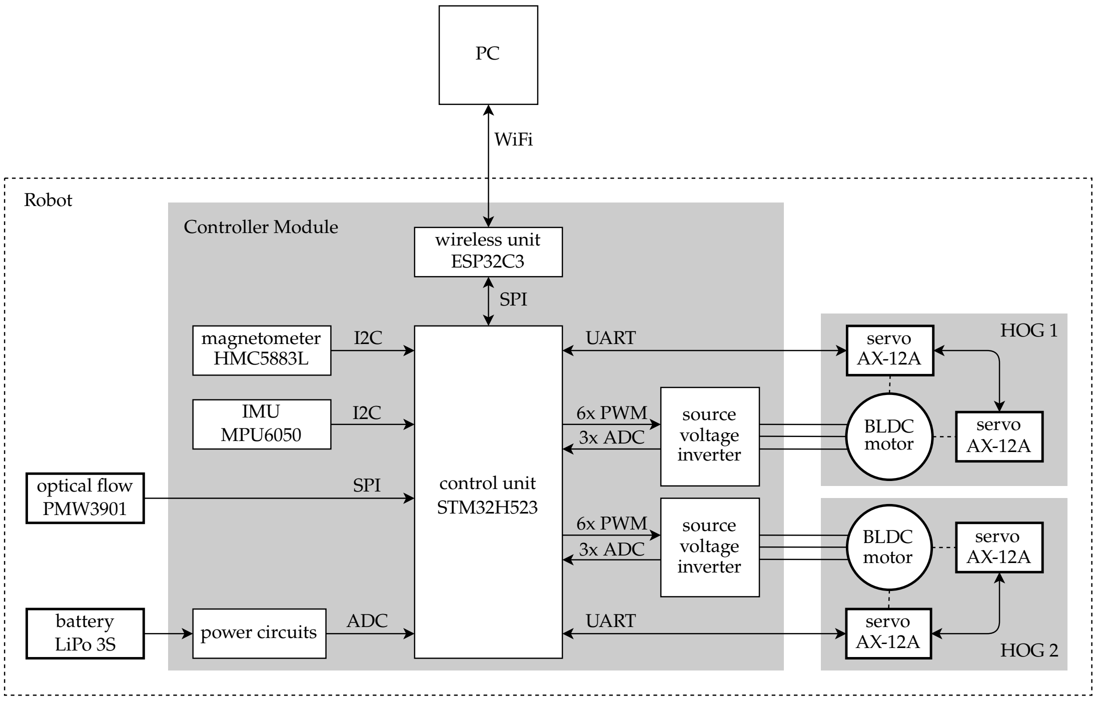


Figure 2.3: Electronic circuits of the *Hogger*^{2} robot

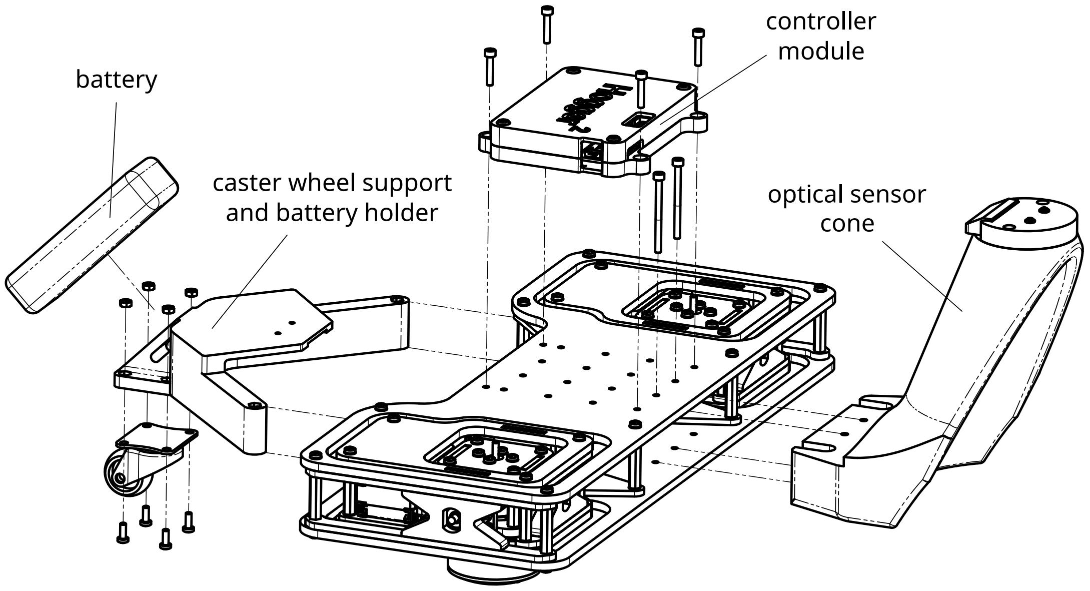


Figure 2.4: Supplemented components

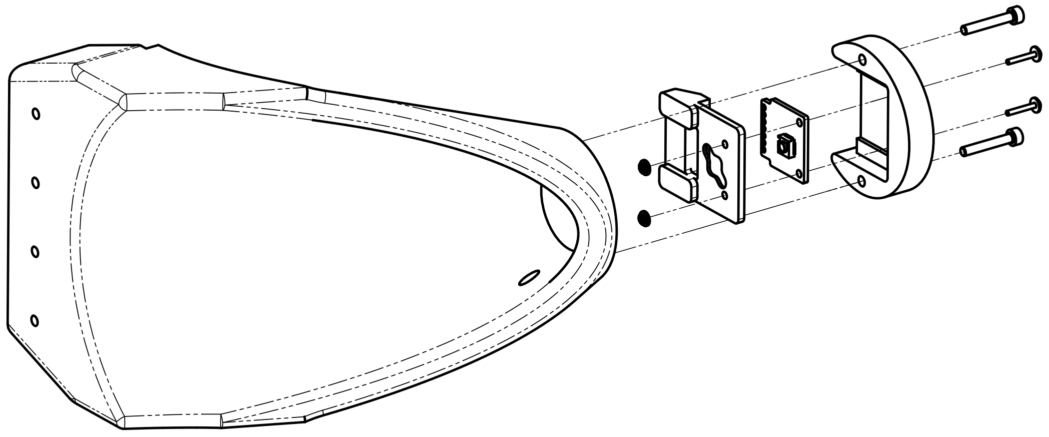


Figure 2.5: Optical flow sensor cone

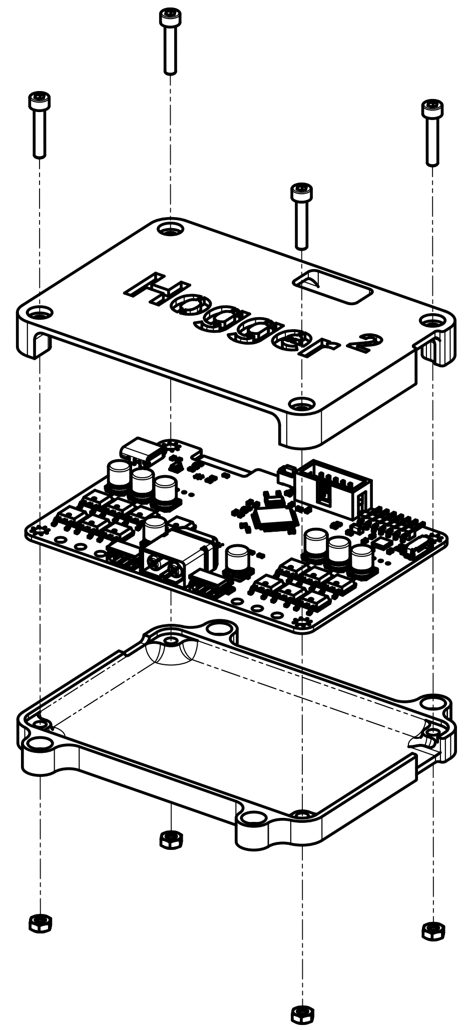


Figure 2.6: Controller module case

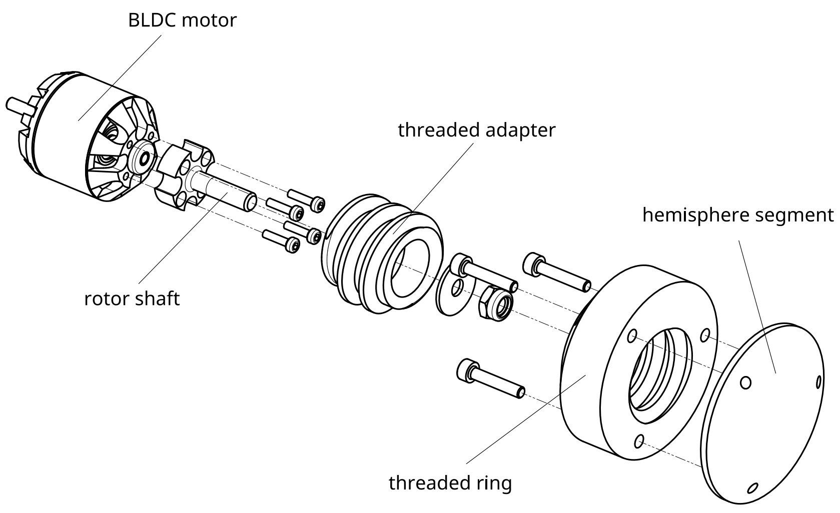


Figure 2.7: Motor–hemisphere connection

ure 2.7, was redesigned to prevent looseness caused by vibrations. The threaded core allows for easy replacement of the wheel itself, enabling tests of how the hemisphere material affects the robot's operation for future applications. Both thread directions are opposite to the desired wheel rotation, so the threads cannot unscrew themselves during normal ground operation. The existing silicone hemispheres were severely damaged at the point coinciding with the axis of rotation, so they were replaced with PLA 3D-printed parts. In summary, the mechanical construction is shown in Figure 2.8.

2.3 BLDC Motor Control

Brushless DC (BLDC) motors, due to their high durability, are the standard for modern robotic structures. Unlike conventional brushed motors, they require dedicated control algorithms to produce rotary motion. This section describes the sensorless control method used for the operation of the *Hogger*^{2} robot.

2.3.1 Source Voltage Inverters

A BLDC motor can be actuated by applying an alternating voltage pattern, commonly called the "six-step" algorithm [LCA] through its phases (U, V, W) via the source voltage inverter circuit shown in Figure 2.9. For practical reasons, the design consists of six separate power switches, arranged into three pairs (half-bridges) of upper (Q_{UH}, Q_{VH}, Q_{WH}) and lower (Q_{UL}, Q_{VL}, Q_{WL}) transistors. Automotive-grade IPD90N04S4L-04 N-channel MOSFETs [Inf] were used, along with LM2105D gate drivers [Ins] (not shown in Figure). Each half-bridge needs to be controlled by a pair of coupled PWM signals, resulting in a total of 12 control variables for both motors. To meet these requirements, the STM32H523 [STM] microcontroller was selected as a main control

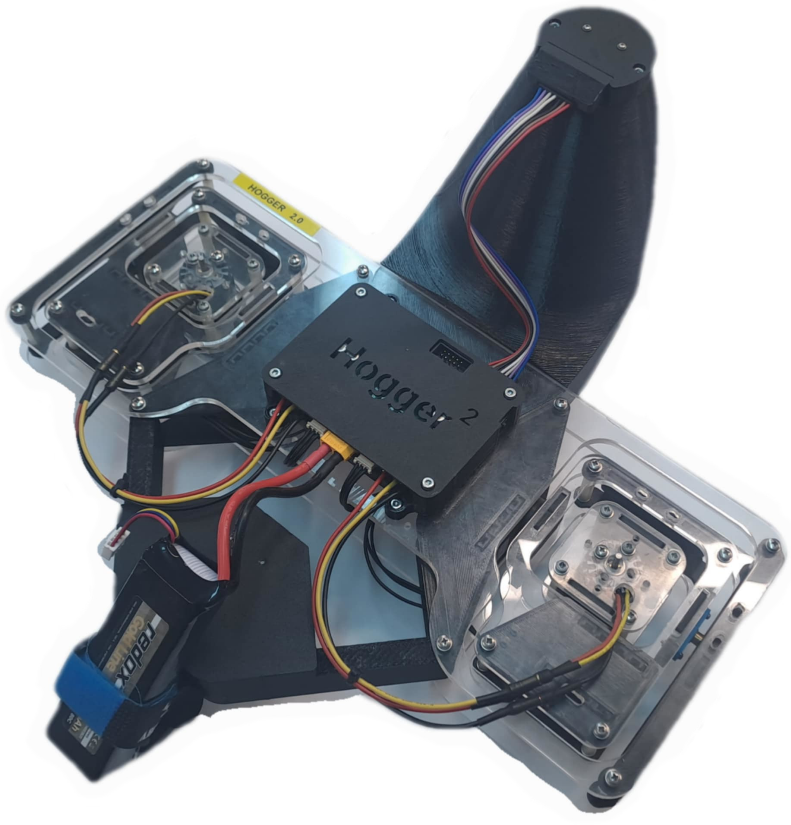


Figure 2.8: The *Hogger*^{2} robot after modifications

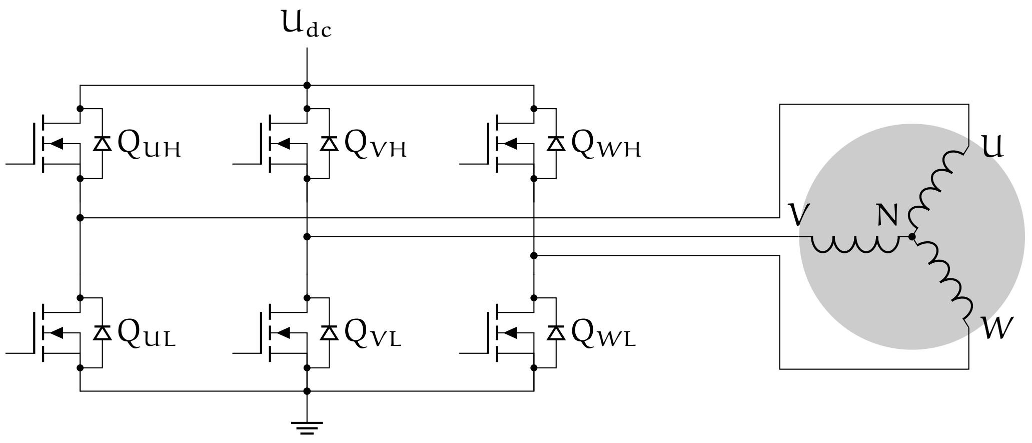


Figure 2.9: Source voltage inverter

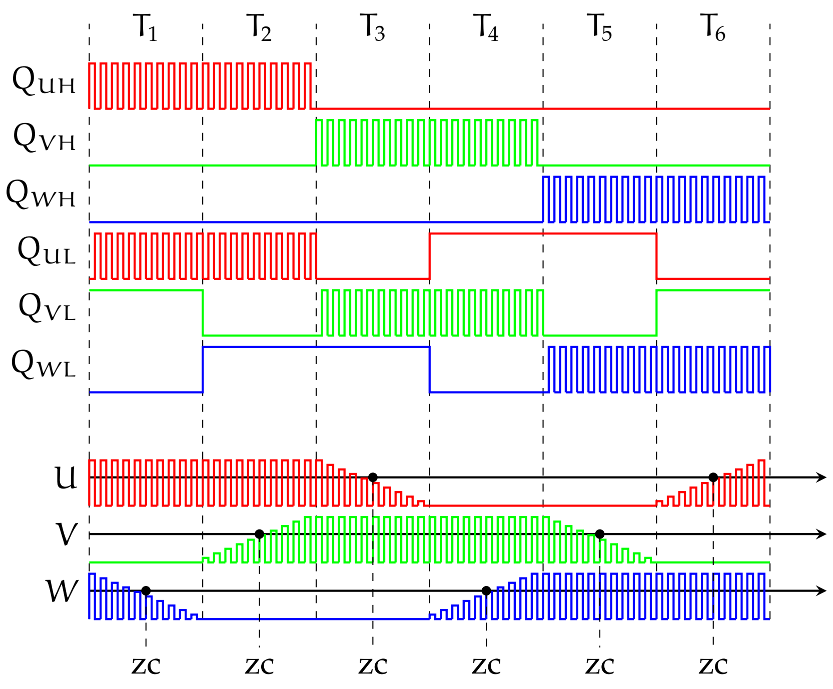


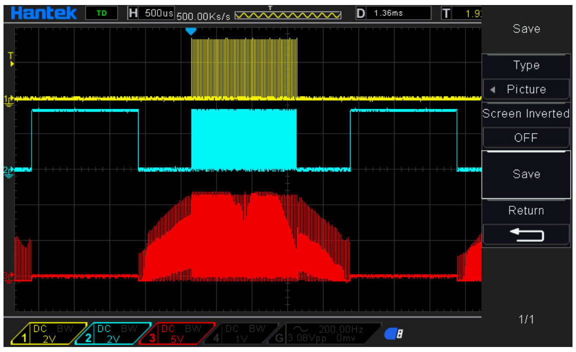
Figure 2.10: Six-step algorithm with back-EMF zero cross events

unit, as it includes two Advanced Control Timer peripherals dedicated to motor control applications.

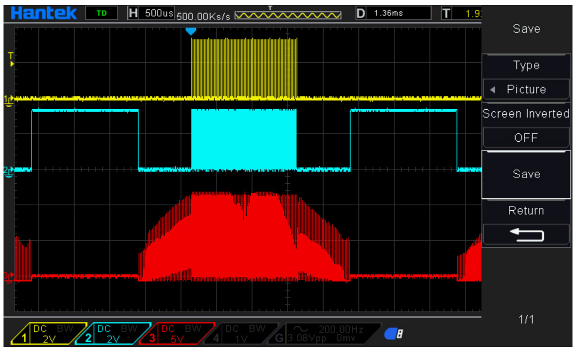
The six-step algorithm, shown in Figure 2.10, considers an ordered sequence of control permutations T for switches Q. To obtain rotary motion, all steps must be applied in the sequence $T_1, T_2, \dots, T_6, T_1, \dots$ for clockwise rotation, or $T_6, T_5, \dots, T_1, T_6, \dots$ for counterclockwise rotation. Each step, also called a commutation, can configure each pair Q_{xH} and Q_{xL} to:

- connect the motor phase to the supply voltage (Q_{xH} high, Q_{xL} low),
- connect the motor phase to the ground (Q_{xH} low, Q_{xL} high),
- left the motor phase floating (Q_{xH} and Q_{xL} low).

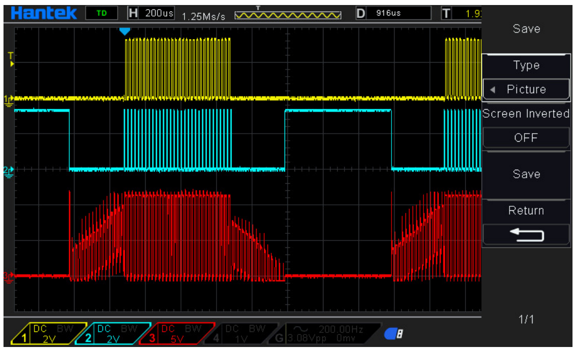
Closing the upper and lower switches of the same half-bridge at the same time creates a short circuit and may damage the transistors. To prevent this situation, an additional 2 µs dead time was configured in the peripheral settings. In each step, one motor phase is powered, one is connected to ground, and the remaining floating phase is used to determine the next step through signal conditioning. For smooth control of the powered phase, 50 kHz center-aligned PWM modulation is applied to the upper and lower switches. Figure 2.11 presents oscillograms of phase and control signals for selected angular velocities. The described method aims to obtain a trapezoidal phase voltage waveform. In practical applications, the minimum angular velocity that the motor controller can handle is 125 $\frac{\text{rad}}{\text{s}}$ without load, and 200 $\frac{\text{rad}}{\text{s}}$ during robot operation. The maximum velocity is constrained by the motor itself. Empirical tests have shown that when the PWM duty cycle exceeds 80% with load, the motor begins to overheat significantly.



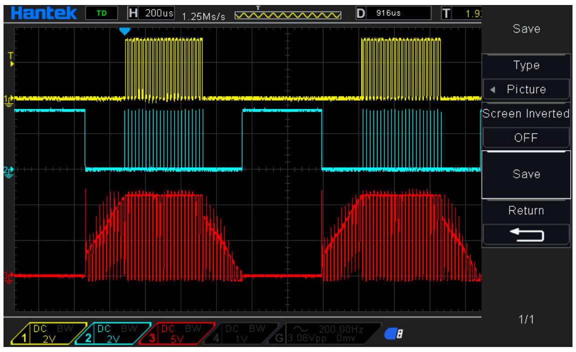
(a) angular velocity 200 $\frac{\text{rad}}{\text{s}}$



(b) angular velocity 250 $\frac{\text{rad}}{\text{s}}$



(c) angular velocity 500 $\frac{\text{rad}}{\text{s}}$



(d) angular velocity 664 $\frac{\text{rad}}{\text{s}}$ (80% duty cycle)

Figure 2.11: Oscillogram of Q_{UH} (yellow), Q_{UL} (blue), and U phase (red)

2.3.2 BEMF Signal Conditioning

Proper motor operation highly depends on the timing of the switching steps in the six-step algorithm. The rotor of the BLDC motor consists of an even number of magnets, forming an alternating magnetic field. By performing steps from T_1 to T_6, the rotor rotates through an angle called an electrical rotation, defined by one pair of magnets. To achieve a full 360° movement, called a mechanical rotation, the six-step algorithm must be performed n times (n electrical rotations), where n is the number of pole pairs in the rotor.

Each step represents $1/6$ of an electrical rotation, which equals 60°. During each step, one winding coil is left floating, and the rotating magnets of the rotor induce a back electromotive force (BEMF) in it. Unfortunately, the BEMF voltage level is proportional to the angular velocity of the rotor, which implies that practical signal conditioning is only possible above a minimum shaft velocity. According to [LCA], the best motor performance is achieved by keeping the point where the BEMF voltage crosses the voltage at the motor's coil interconnection (point N in Figure 2.9) in offset of 30° from the beginning of each step. This point is called the zero-crossing event (ZC) and its precise localization is the greatest difficulty during implementation of the algorithm.

The BEMF signal can be observed from the U, V, and W phases by the control unit using ADC through resistor dividers and small filtering capacitors. The neutral voltage

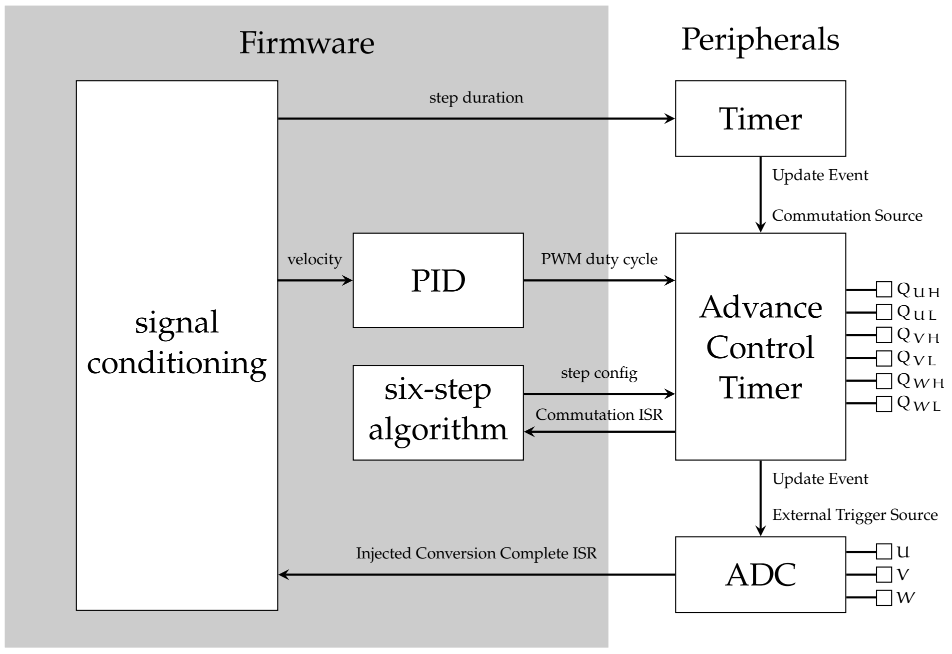


Figure 2.12: Controller implementation on the STM32H5

at point N can be calculated as the average of phase voltages

$$U_N = \frac{U + V + W}{3}.$$

Controller implementation is presented in Figure 2.12. The STM32 ADC peripheral was synchronized with Advanced Control Timer to perform conversion in the middle of PWM high period using “Injected Conversion” feature. Then, on each PWM period, voltage from appropriate floating phase is compared with U_N voltage by the software. To prevent false positive detections, the majority filter described in [LCA] was implemented. Result of each comparison is stored in the 8-bit variable acting as a shift register. If the number of positive comparisons (ones) is greater than the predefined threshold k_{th}, then the ZC event occurs roughly $\frac{1}{2}k_{th}T_{PWM}$ before the conversion starts. Using this knowledge, it is possible to calculate the time of next commutation in the sequence.

Unfortunately, under high motor load fluctuations experienced during movement using hemispheres, additional filtering was needed. In summary, period t_{i+1} of the next commutation T_{i+1} can be calculated as

$$\begin{cases} t_{izc} = t_{conv} - \frac{1}{2}k_{th}T_{PWM} \\ t_{i+1ideal} = 2t_{izc} \\ t_{i+1} = \alpha_t t_{i+1ideal} + (1-\alpha_t)t_i \end{cases},$$

where t_{conv} is conversion start time (counted from the beginning of the step T_i), t_{izc} is ZC event occur time, $t_{i+1ideal}$ is theoretical duration of next commutation, and t_{i+1} is duration that is applied in reality. For motors KAVAN 2830 750 KV parameters $k_{th} = 4$, $\alpha_t = 0.05$ were found empirically.

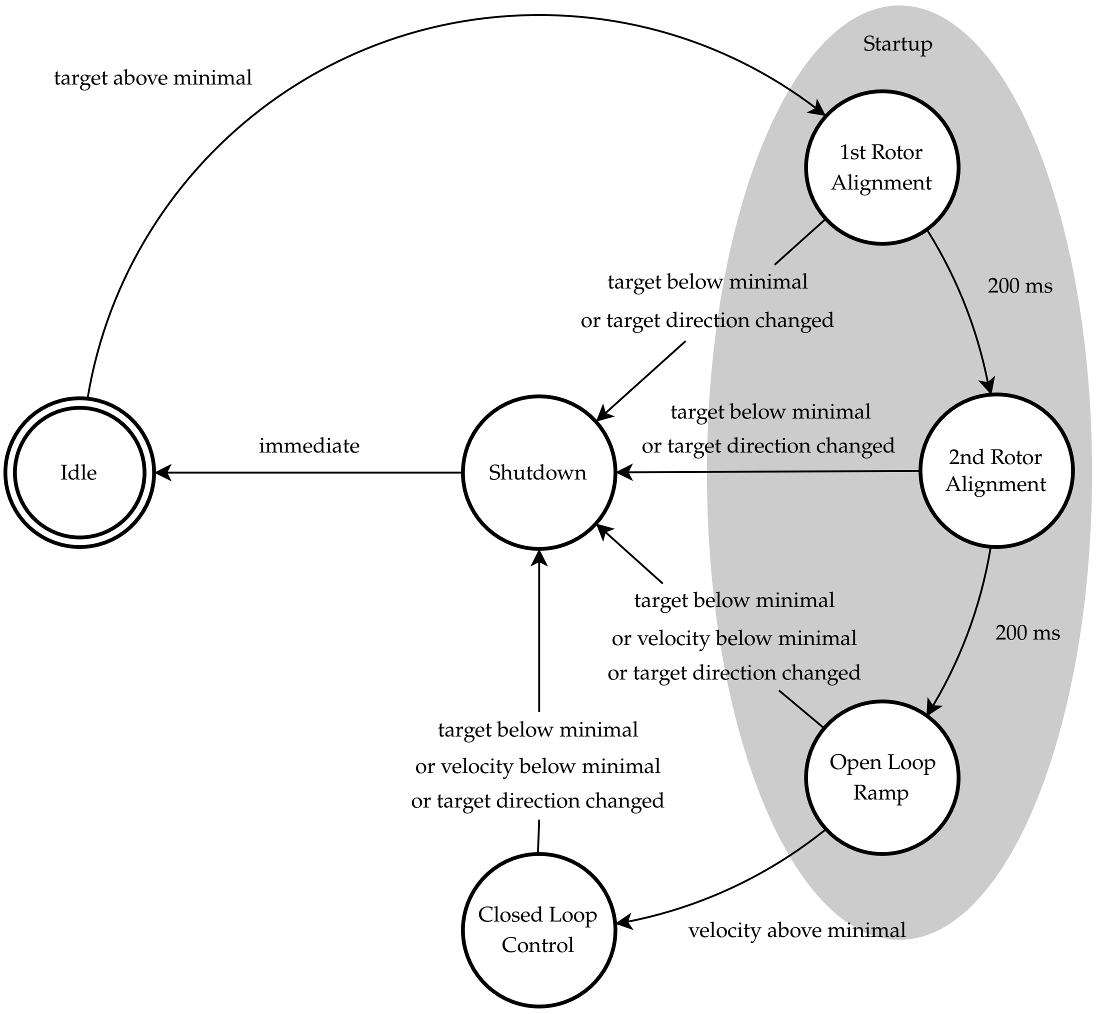

Startup
target above minimal
1st Rotor Alignment
200 ms
target below minimal or target direction changed
target below minimal or target direction changed
Idle
immediate
Shutdown
2nd Rotor Alignment
200 ms
target below minimal or velocity below minimal or target direction changed
target below minimal or velocity below minimal or target direction changed
Open Loop Ramp
Closed Loop Control
velocity above minimal

Figure 2.13: Motor controller state machine

Mechanical angular velocity of the motor shaft ω can be calculated and filtered by

$$\begin{cases} \omega_{zc} = 2\pi \dfrac{n_{zc}}{6pT_{zc}} \\ \omega_{i+1} = \alpha_\omega \omega_{zc} + (1-\alpha_\omega)\omega_i \end{cases},$$

where ω_{zc} is instantaneous angular velocity, ω_i is filtered angular velocity, p is number of rotor pole pairs, and n_{zc} is number of ZC events observed in time T_{zc}. For the selected motors $p = 7$, and empirical tests shown that parameters $T_{zc} = 20$ ms, $\alpha_\omega = 0.1$ are giving sufficiently good performance in practice.

2.3.3 Startup and Closed-Loop Operation

The motor controller logic, presented in Figure 2.13, is defined by a finite state machine that handles motor startup, operation, and shutdown. The startup procedure consists of rotor alignment phases followed by a velocity ramp controlled in open-loop mode. The first rotor alignment stage sets the rotor to a known position by applying predefined commutation T_1 for a specified duration. The second alignment ensures that

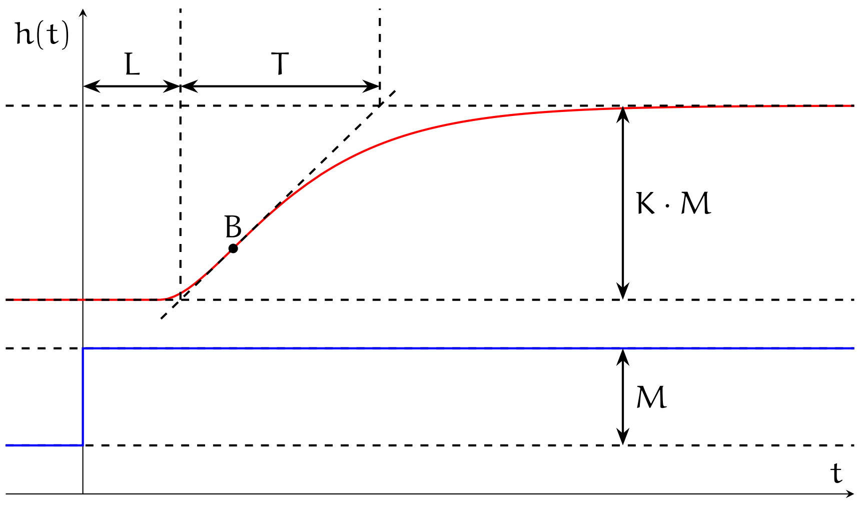


Figure 2.14: Geometric properties of the step response of an inertial system

the rotor spins in the correct direction by applying commutation T_1 or T_5, according to the desired direction. Both are performed using a PWM duty cycle of $u = 30\%$ over a 200 ms period.

When alignment is complete, a startup ramp is applied to accelerate the rotor to a point where BEMF signal conditioning becomes feasible. Subsequent commutations are applied to the inverter circuit, each with increasing frequency f and a constant PWM duty cycle of $u = 40\%$. The ramp curve is defined as

$$f(t) = f_{max}\left(1 - e^{-\lambda t}\right),$$

where $f_{\text{max}} = 1$ kHz is the maximum commutation frequency, $\lambda = 1$ is the acceleration parameter, and t is the time elapsed since the beginning of the ramp. Empirical tests have shown that the proposed ramp provides more robust results than linear ramps and that the correct ramp shape is key to a successful startup. To ensure a smooth transition between open-loop and closed-loop control, a special blending function is used to merge both signals over a 1 second period after the ramp is complete.

The closed-loop operation is performed using a PID controller to maintain the motor's angular velocity at the target value, using the PWM duty cycle u as the control input. For each motor, a PID controller was tuned using the Chien-Hrones-Reswick (CHR) method with 20% overshoot [ÅH95]. The results of the tuning procedures are included in Appendix B. The method, shown in Figure 2.14, involves drawing additional straight lines onto the step response (red plot) of an unknown inertial system, representing:

- the signal level at steady state before applying the step,
- the signal level at steady state after applying the step,
- the tangent line with the maximum slope (at point B).

Analysis of the line intersections reveals the basic geometric properties K (gain), T (time constant), and L (lag) of the simplified system, which approximates the plant.

These properties can be used to calculate PID parameters that ensure good closed-loop performance using the CHR 20% overshoot formulas

$$\begin{cases} K_p = 0.95\dfrac{T}{KL} \\ K_i = 0.68\dfrac{1}{KL} \\ K_d = 0.45\dfrac{T}{K} \end{cases} .$$

To automate the process of obtaining PID settings from the step response, a Python script was created to find the parameters of the best-fitting four-parameter model in the form

$$H_m(s) = \frac{K_m}{(T_{1m}s+1)(T_{2m}s+1)}e^{-L_m s},$$

which step response in the time domain is given by

$$h_m(t) = \begin{cases} 0 & t \leqslant L_m \\ K_m\left(1 + \dfrac{T_{2m}e^{-\frac{t-L_m}{T_{2m}}} - T_{1m}e^{-\frac{t-L_m}{T_{1m}}}}{T_{1m}-T_{2m}}\right) & t > L_m \end{cases},$$

where T_{1m} and T_{2m} are time constants, L_m is a delay, and K_m is the model gain. Using this model, it is possible to algebraically determine the point B, and ultimately extract the K, T, and L properties.

2.4 Sensor Fusion

Reliable knowledge about robot state is a key to precise control. The robot x, y, θ coordinates are not possible to measure directly without external reference systems like GNSS or radio anchors [KCB+23]. Sensor fusion techniques allows for obtaining information about the system state from multiple measurement sources which is more reliable than each of the measurements separately. In this work an Extended Kalman Filter has been used for this purpose.

2.4.1 Extended Kalman Filter

In order to estimate the 2D pose of the robot using only on-board sensors, Extended Kalman Filter (EKF) [Wika] was developed. EKF is a sensor fusion technique that enables the estimation $\hat{\mathbf{x}}$ of the unknown state $\mathbf{x}$ of the discrete system in the form

$$\begin{cases} \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \boldsymbol{\omega}_{k-1} \\ \mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \boldsymbol{\nu}_k \end{cases},$$

where $\mathbf{u}$ are the system inputs, $\mathbf{z}$ are the measurements, and $\boldsymbol{\omega}$ and $\boldsymbol{\nu}$ are process and observation noises, both assumed to be zero-mean Gaussian. In the following description it is assumed that the system operates at fixed time intervals Δ_t between time steps k.

The algorithm consists of two steps: prediction and correction. The prediction step can be defined as

$$\begin{cases} \hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\mathbf{x}_{k-1|k-1}, \mathbf{u}_{k-1}) \\ \hat{\mathbf{P}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{P}}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_{k-1} \end{cases},$$

where

$$\mathbf{F}_k = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k},$$

$\mathbf{f}$ is the system model, $\mathbf{Q}$ is the process noise covariance matrix, and $\hat{\mathbf{P}}$ is the estimate of state vector covariance matrix. The notation $x_{n|m}$ represents the value of x at time n given observations from time periods where $m \leqslant n$. In this step, the input vector $\mathbf{u}$ can be obtained from sensors recording of relatively high-frequencies. It is crucial to perform the prediction step at fixed time intervals. Unfortunately, this implies prediction frequency to be equal to the sampling frequency of the slowest predicting sensor.

The correction step can be defined as

$$\begin{cases} \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})) \\ \hat{\mathbf{P}}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \hat{\mathbf{P}}_{k|k-1} \end{cases},$$

where

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_k (\mathbf{H}_k \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1},$$

$$\mathbf{H}_k = \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k|k-1}},$$

$\mathbf{h}$ is the sensor model, and $\mathbf{R}$ is the sensor noise covariance matrix. As opposed to the prediction step, the correction step can be performed separately for each correcting sensor. This effectively enables operation at the full sampling frequency of each sensor. Correcting measurements represent relatively low-frequency signals.

2.4.2 Estimator Implementation

For the *Hogger*[2] robot, the state vector $\mathbf{x}$ we will define as

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} & \theta & \dot{\mathbf{p}} & \dot{\theta} & \gamma_m \end{bmatrix}^T = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} & \dot{\theta} & \gamma_m \end{bmatrix}^T,$$

where $\mathbf{p}$ with components x, y represents the robot's position, and θ represents the robot's orientation. Apart from that, due to the practical reasons it is common to incorporate environment depended system model parameters into the state vector itself. Here such an element is γ_m that represents the angle of magnetic inclination related to the magnetometer sensor model (described in paragraph below "Magnetometer model").

System Model

The system model consider inputs $\mathbf{u}$ in the form

$$\mathbf{u} = \begin{bmatrix} a_x & a_y & \omega_z & \dot{\omega}_z \end{bmatrix}^T,$$

where ω_z is gyroscope measurement of robot's angular velocity, and $\dot{\omega}_z$ is angular acceleration computed numerically. Terms a_x, a_y are linear components of acceleration vector $\mathbf{a}$ observed in accelerometer's frame located with offset $\mathbf{r}_a$

$$\mathbf{r}_a = \begin{bmatrix} r_{ax} & r_{ay} \end{bmatrix}^T.$$

from the robot's frame origin. Acceleration and angular velocity measurements comes from MPU6050 IMU sensor [Inv], with sampling frequency 1 kHz.

To model the system, rotational movement can be approximated as a uniform rotational motion in the form of

$$\theta_k = \theta_{k-1} + \Delta_t \dot{\theta}_{k-1},$$

where Δ_t is a short time interval equal to IMU sampling period. Analogously, position $\mathbf{p}$ can be approximated using uniformly accelerated motion in the form of

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_{k-1} + \Delta_t \dot{\mathbf{p}}_{k-1} + \frac{1}{2}\Delta_t^2 \ddot{\mathbf{p}}_{k-1}, \\ \dot{\mathbf{p}}_k &= \dot{\mathbf{p}}_{k-1} + \Delta_t \ddot{\mathbf{p}}_{k-1}, \end{aligned}$$

where $\ddot{\mathbf{p}}$ is the robot's acceleration expressed in the world's reference frame. An accelerometer mounted on the rotating body senses acceleration $\mathbf{a}$ expressed as

$$\mathbf{a} = \mathbf{a}_b + \mathbf{a}_c + \mathbf{a}_t,$$

where

$$\begin{cases} \mathbf{a}_c = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}_a) \\ \mathbf{a}_t = \boldsymbol{\alpha} \times \mathbf{r}_a \end{cases},$$

$\mathbf{a}_b$ is acceleration of the body itself, $\mathbf{a}_c$ is centripetal acceleration due to rotational movement $\boldsymbol{\omega} = [0\ 0\ \dot{\theta}]^T$, and $\mathbf{a}_t$ is tangential acceleration due to presents of angular acceleration $\boldsymbol{\alpha} = [0\ 0\ \dot{\omega}_z]^T$. All of them are expressed in the accelerometer's reference frame. The robot's acceleration in the world's frame can be defined as

$$\ddot{\mathbf{p}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a_{bx} \\ a_{by} \end{bmatrix}.$$

In summary, the system model $\mathbf{f}$ is defined as

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} x + \Delta_t \dot{x} + \frac{1}{2}\Delta_t^2 \left(a_{bx}\cos(\theta) - a_{by}\sin(\theta)\right) \\ y + \Delta_t \dot{y} + \frac{1}{2}\Delta_t^2 \left(a_{bx}\sin(\theta) + a_{by}\cos(\theta)\right) \\ \theta + \Delta_t \dot{\theta} \\ \dot{x} + \Delta_t \left(a_{bx}\cos(\theta) - a_{by}\sin(\theta)\right) \\ \dot{y} + \Delta_t \left(a_{bx}\sin(\theta) + a_{by}\cos(\theta)\right) \\ \omega_z \\ \gamma_m \end{bmatrix},$$

where $a_{bx} = a_x + r_{ax}\dot{\theta}^2 + \dot{\omega}_z r_{ay}$, and $a_{by} = a_y + r_{ay}\dot{\theta}^2 - \dot{\omega}_z r_{ax}$.

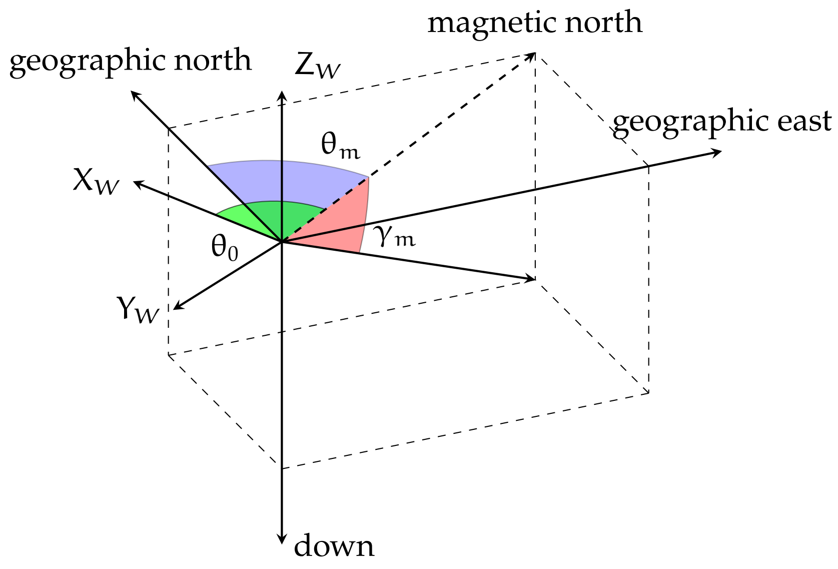


Figure 2.15: Earth's magnetic field direction

Magnetometer Model

Relying only on dead-reckoning using gyroscope measurements is not robust enough to obtain reliable information about the robot's orientation. Sensor HMC5883L [Hon], with a sampling frequency 200 Hz, was introduced as a magnetic north orientation reference.

Earth's magnetic field, illustrated in Figure 2.15, is not identical at every point on the globe and it has two properties: the declination angle θ_m and the inclination angle γ_m. Magnetic declination θ_m is the deviation angle from the true geographic north. During the robot operation, it is helpful to set an artificial declination θ_0 (see Figure 2.15) to align the world's reference frame $X_W Y_W Z_W$ with the environment (e.g., building walls). Magnetic inclination γ_m is the deviation angle from the horizontal plane. Estimation of inclination allows the system to adapt to the changing environment in which the robot operates. Using this knowledge, the magnetometer model can be expressed as

$$\mathbf{h}_{mag}(\mathbf{x}) = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} \cos(\theta+\theta_0) & \sin(\theta+\theta_0) & 0 \\ -\sin(\theta+\theta_0) & \cos(\theta+\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\gamma_m \\ 0 \\ \sin\gamma_m \end{bmatrix},$$

where m_x, m_y and m_z are components of the normalized magnetic field vector.

Optical Flow Sensor Model

Velocity correction is performed using the optical motion sensor PMW3901 [Pix], commonly called "optical flow" sensor. The sensor is capable of measuring Δp_x and Δp_y pixel shifts between frames with a 50 Hz sampling frequency. Measurements are expressed in the camera frame, which is located at an offset $\mathbf{r}_f$

$$\mathbf{r}_f = \begin{bmatrix} r_{fx} & r_{fy} \end{bmatrix}^T$$

from the robot's frame origin. Using the technique described in [Jac21] it is possible to define the relation between robot's velocities and $\Delta \mathbf{p}$ as

$$h_{flow}(\mathbf{x}) = \begin{bmatrix} \Delta p_x \\ \Delta p_y \end{bmatrix} = \Delta_{tf} \frac{f_f}{h_f} \left(\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} -r_{fy}\dot{\theta} \\ r_{fx}\dot{\theta} \end{bmatrix} \right),$$

where f_f is the focal length of the lens, h_f is the sensor's height above the ground, and Δ_{tf} is sensor sampling period.

2.4.3 Sensor Calibration

Measurement models from the previous section describe the behavior of ideal sensors. Unfortunately, this does not hold for the low-cost MEMS elements used in this system. This section describes techniques to increase the reliability of these measurements.

Gyroscope Calibration

Simple MEMS gyroscopes are susceptible to bias drift (also known as zero-drift phenomenon). Temperature changes cause unwanted variations in the gyroscope's output, even when it is not experiencing actual angular motion. To compensate for this drift, the raw measurement values ω_{raw} for each axis are averaged over a stationary period of approximately 10 seconds. The resulting bias is then subtracted from the raw values, as shown below:

$$\omega_{comp} = \omega_{raw} - \overline{\omega}_{raw}.$$

This calibration procedure should be performed before each experimental session.

Accelerometer Calibration

As with the gyroscope, the zero-drift phenomenon is also present during the operation of the accelerometer. To further increase the reliability of the measurements, a full 3D rotation, translation, and scaling can be applied to the raw sensor outputs as follows:

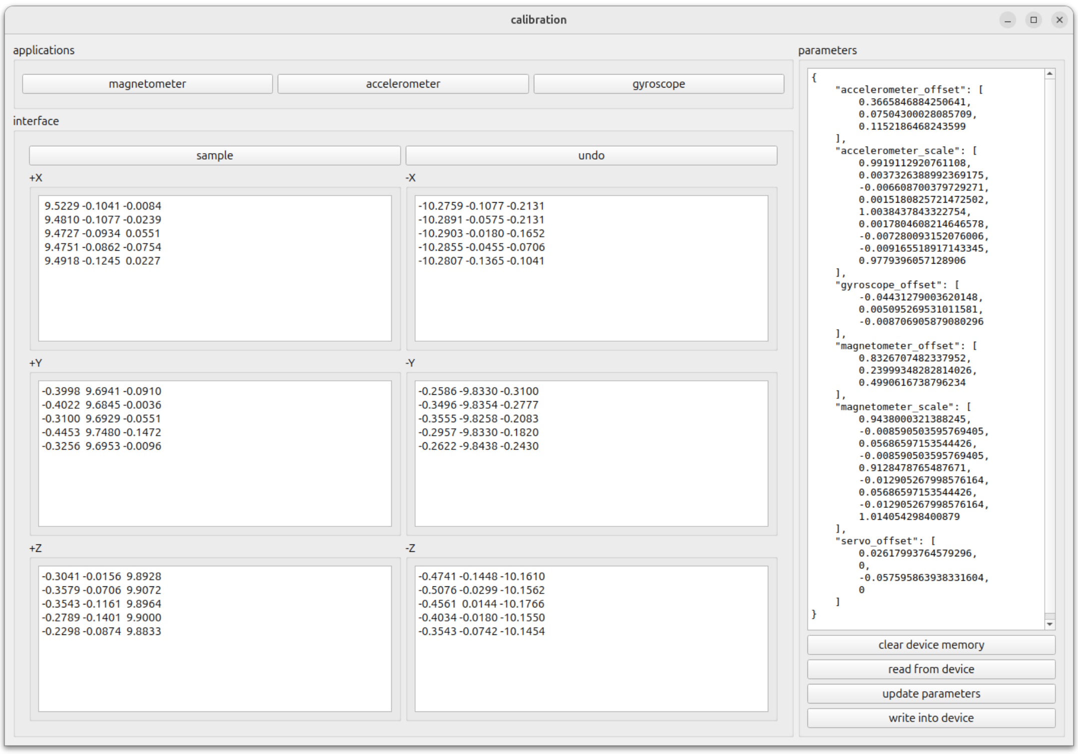
$$\mathbf{a}_{comp} = \mathbf{M}_a \mathbf{a}_{raw} + \mathbf{b}_a,$$

where $\mathbf{M}_a$ is the combined rotation and scale matrix, and $\mathbf{b}_a$ is the offset vector. Both parameters can be obtained using the least-squares method

$$\begin{bmatrix} \mathbf{M}_a^T \\ \mathbf{b}_a^T \end{bmatrix} = \left(J_a^T J_a\right)^{-1} J_a^T K,$$

where

$$J_a = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & 1 \end{bmatrix}, \qquad K = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \\ \vdots & \vdots & \vdots \\ X_N & Y_N & Z_N \end{bmatrix},$$



+X
```
9.5229 -0.1041 -0.0084
9.4810 -0.1077 -0.0239
9.4727 -0.0934  0.0551
9.4751 -0.0862 -0.0754
9.4918 -0.1245  0.0227
```

-X
```
-10.2759 -0.1077 -0.2131
-10.2891 -0.0575 -0.2131
-10.2903 -0.0180 -0.1652
-10.2855 -0.0455 -0.0706
-10.2807 -0.1365 -0.1041
```

+Y
```
-0.3998  9.6941 -0.0910
-0.4022  9.6845 -0.0036
-0.3100  9.6929 -0.0551
-0.4453  9.7480 -0.1472
-0.3256  9.6953 -0.0096
```

-Y
```
-0.2586 -9.8330 -0.3100
-0.3496 -9.8354 -0.2777
-0.3555 -9.8258 -0.2083
-0.2957 -9.8330 -0.1820
-0.2622 -9.8438 -0.2430
```

+Z
```
-0.3041 -0.0156  9.8928
-0.3579 -0.0706  9.9072
-0.3543 -0.1161  9.8964
-0.2789 -0.1401  9.9000
-0.2298 -0.0874  9.8833
```

-Z
```
-0.4741 -0.1448 -10.1610
-0.5076 -0.0299 -10.1562
-0.4561  0.0144 -10.1766
-0.4034 -0.0180 -10.1550
-0.3543 -0.0742 -10.1454
```

```
{
    "accelerometer_offset": [
        0.3665846884250641,
        0.07504300028085709,
        0.1152186468243599
    ],
    "accelerometer_scale": [
        0.9919112920761108,
        0.0037326388992369175,
        -0.006608700379729271,
        0.0015180825721472502,
        1.0038437843322754,
        0.0017804608214646578,
        -0.007280093152076006,
        -0.009165518917143345,
        0.9779396057128906
    ],
    "gyroscope_offset": [
        -0.04431279003620148,
        0.005095269531011581,
        -0.008706905879080296
    ],
    "magnetometer_offset": [
        0.8326707482337952,
        0.23999348282814026,
        0.4990616738796234
    ],
    "magnetometer_scale": [
        0.9438000321388245,
        -0.008590503595769405,
        0.05686597153544426,
        -0.008590503595769405,
        0.9128478765487671,
        -0.012905267998576164,
        0.05686597153544426,
        -0.012905267998576164,
        1.014054298400879
    ],
    "servo_offset": [
        0.02617993764579296,
        0,
        -0.057595863938331604,
        0
    ]
}
```

Figure 2.16: Accelerometer calibration app

x_i, y_i, and z_i are the elements of the i-th $\mathbf{a}_{raw}$ sample, X_i, Y_i, and Z_i are the elements of the corresponding reference acceleration vector $\mathbf{a}_{ref}$, and N is the number of collected samples. The simplest reference accelerations can be obtained by using gravity:

$$\mathbf{a}_{ref} \in \left\{ \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \begin{bmatrix} -g \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \right\},$$

where g is a value of Earth's gravity acceleration.

For practical reasons, a dedicated PC application for performing sensor calibration, with interface shown in Figure 2.16, was created using the Qt6 [Gro] and Eigen [JG] libraries in C++. The procedure involves placing the robot in six different orientations using, e.g., flat mechanical fixtures on a level surface. Typically, collecting three to five samples in each reference direction is sufficient to achieve satisfactory calibration results.

Magnetometer Calibration

A magnetometer placed inside the robot's frame senses a distorted magnetic field. According to [Ozy], there are two kinds of magnetic distortion: soft-iron and hard-iron distortions. Soft-iron distortions are related to scaling of the resulting vector. Hard-iron distortions offset all measurements by a constant vector. In practical applications, it is necessary to compensate for the influence of both of these phenomena. Translation and scaling can be applied to the raw sensor outputs in the following way:

$$\mathbf{m}_{comp} = \mathbf{M}_m \mathbf{m}_{raw} + \mathbf{b}_m,$$

where $\mathbf{M}_m$ is the combined rotation and scale matrix, and $\mathbf{b}_m$ is the offset vector. Both can be obtained by collecting a dataset of raw measurements and fitting an ellipsoid to it in standard form

$$p_1x^2 + p_2y^2 + p_3z^2 + p_4xy + p_5yz + p_6zx + p_7x + p_8y + p_9z + 1 = 0,$$

where $p_i, i = 1, \dots, 9$ creates the ellipsoid coefficients vector $\mathbf{p}$. Thus, the above equation can be rewritten into

$$\mathbf{J}_m\mathbf{p} = 0,$$

where

$$\mathbf{J}_m = \begin{bmatrix} x_1^2 & y_1^2 & z_1^2 & x_1y_1 & y_1z_1 & z_1x_1 & x_1 & y_1 & z_1 & 1 \\ x_2^2 & y_2^2 & z_2^2 & x_2y_2 & y_2z_2 & z_2x_2 & x_2 & y_2 & z_2 & 1 \\ x_3^2 & y_3^2 & z_3^2 & x_3y_3 & y_3z_3 & z_3x_3 & x_3 & y_3 & z_3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & y_N^2 & z_N^2 & x_Ny_N & y_Nz_N & z_Nx_N & x_N & y_N & z_N & 1 \end{bmatrix},$$

with x_i, y_i, and z_i being the elements of the i-th $\mathbf{m}_{raw}$ sample, and N the number of collected samples in the dataset. To achieve good fitting performance, Singular Value Decomposition (SVD) [Wikc] was used to decompose $\mathbf{J}_m$ into three matrices

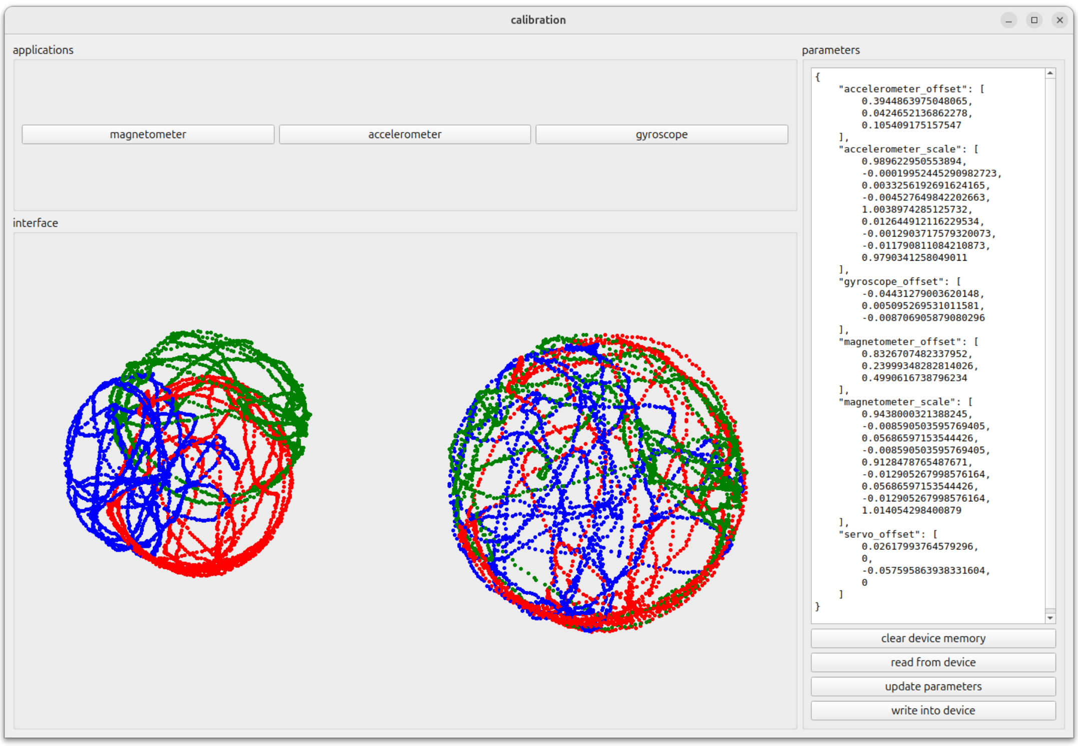
$$\mathbf{J}_m = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{U}$ is the matrix of left singular vectors, $\mathbf{\Sigma}$ is the diagonal matrix of singular values, and $\mathbf{V}$ is the matrix of right singular vectors. The null-space solution $\mathbf{p}$ is given by the singular vector corresponding to the smallest singular value, which is usually the last column of $\mathbf{V}$.

Like for the other sensors, the PC calibration app contains a magnetometer calibration feature, as shown in Figure 2.17. To simplify the interface, magnetometer samples are presented as three planar projections: XY (red), YZ (green), and ZX (blue). The left plot shows the samples before calibration (dataset), while the right plot visualizes the data after transformation. Both charts are updated in real time for every telemetry frame received from the robot. The calibration procedure involves rotating the robot in space until the points of different colors on the right plot overlap. This indicates that the transformation has converged and the calibrated samples lie on a sphere.

2.5 Communication

Reliable communication with robot subsystems is an important requirement during debugging and supervision, even for autonomous robots. The *Hogger*[2] robot is equipped with an ESP32C3 microcontroller to ensure wireless communication between the main control unit (STM32) and a PC. Figure 2.18 presents the STM32-PC communication channel. In [Lub23] it is proposed to apply a JSON-based communication



```
{
    "accelerometer_offset": [
        0.3944863975048065,
        0.0424652136862278,
        0.105409175157547
    ],
    "accelerometer_scale": [
        0.989622950553894,
        -0.00019952445290982723,
        0.0033256192691624165,
        -0.004527649842202663,
        1.0038974285125732,
        0.012644912116229534,
        -0.0012903717579320073,
        -0.011790811084210873,
        0.9790341258049011
    ],
    "gyroscope_offset": [
        -0.04431279003620148,
        0.005095269531011581,
        -0.008706905879080296
    ],
    "magnetometer_offset": [
        0.8326707482337952,
        0.23999348282814026,
        0.4990616738796234
    ],
    "magnetometer_scale": [
        0.9438000321388245,
        -0.008590503595769405,
        0.05686597153544426,
        -0.008590503595769405,
        0.9128478765487671,
        -0.012905267998576164,
        0.05686597153544426,
        -0.012905267998576164,
        1.014054298400879
    ],
    "servo_offset": [
        0.02617993764579296,
        0,
        -0.057595863938331604,
        0
    ]
}
```

Figure 2.17: Magnetometer calibration app

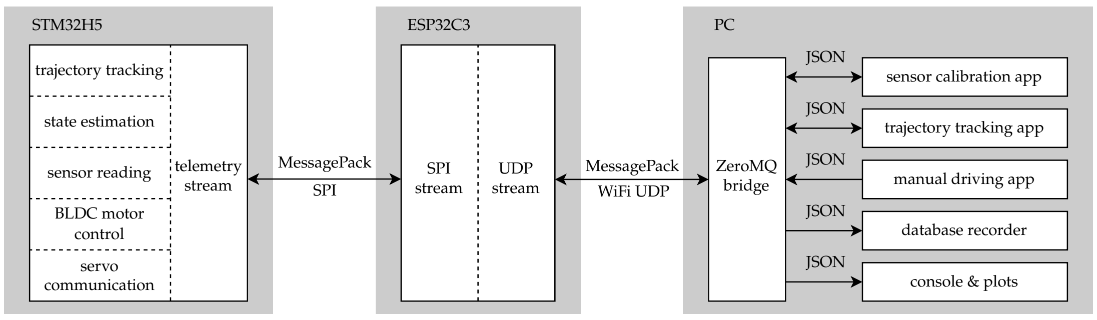


Figure 2.18: Robot communication channel

protocol [Wikb]. This solution allows sending messages with a complex and nested structure without the need for schema files. To further increase bandwidth, it was decided to use the MessagePack [Fur] format, which is used as a binary equivalent to JSON.

2.5.1 Connection Between Main and Network Microcontrollers

The main microcontroller is connected to the network microcontroller using the 4-wire SPI bus, where the STM32 is the master and the ESP32 is the slave. In addition, the *HANDSHAKE* signal is used to indicate the slave's readiness for the next SPI transaction. The transactions, each with a size of 2 kB in both directions, are performed continuously and consist of:

- 2-byte header – number of valid bytes in the payload,
- 2046 bytes of payload.

This structure ensures that the transaction size is constant, greatly simplifying the logic on both sides and enabling DMA support. The bus clock is set to 31.25 Mbps, exceeding the capabilities of the WiFi driver on the ESP32, which claims to support up to 30 Mbps of UDP throughput under ideal conditions. MessagePack messages are parsed using the cmp library [Gun] for the C programming language.

2.5.2 Communication Between Robot and Station

The on-board ESP32 acts as an intermediary between the robot's logic (STM32) and a PC. The ESP32 works as a WiFi station and connects to a nearby network with the predefined SSID. Data received over SPI is redirected to the predefined network socket, and vice versa. For further increase channel bandwidth, UDP transmission stream is used in both directions.

In case of connection loss with the station, the ESP sends a special stop message to the host to ensure that the robot's actuators will halt. Conversely, if a timeout occurs on the host, the ESP informs the station about the error.

2.5.3 ZeroMQ Bridge

Endpoint apps running on the station PC can send and receive data to and from the robot through a dedicated ZeroMQ bridge [zer]. The major purposes of the bridge are:

- maintaining network connection with a robot,
- reading and writing data to ZeroMQ network,
- converting messages between JSON and MessagePack formats,
- collecting throughput statistics.

This solution allows multiple dedicated apps to run in parallel for plotting or sending commands, while maintaining messages in a human-readable form. ZeroMQ libraries are available in many programming languages, so endpoint apps can be written in languages such as Python or C++.

Chapter 3

Control Algorithms

Due to its unconventional drive system, the *Hogger*^{2} robot still does not have well-tested control methods. In recent years, simulation research took place as part of several theses [Jon17, Boc19]. This chapter describes previously designed control algorithms as well as new approaches to controlling the *Hogger*^{2} robot.

All algorithms consider robot's reference frame and gimbal angles like shown in Figure 3.1. Origin of the robot's reference frame $X_B Y_B Z_B$ is placed in the point of two gimbal axes ϕ_1 and θ_1 crossing. The positive end of the X_B axis is directed toward the front of the robot, while the positive end of the Y_B axis represents the left side of the robot's body.

3.1 Simultaneous Algorithm

The simultaneous algorithm, originally introduced in [Jon17], assumes same control signals for both HOG wheels. Additionally, it is assumed that the gimbal movement of the hemisphere has no influence on the robot's body movement. The system can be expressed as

$$\begin{cases} \dot{x} = R\dot{\psi}_u \left(\cos\theta \cos\theta_u \sin\phi_u + \sin\theta \sin\theta_u\right) \\ \dot{y} = R\dot{\psi}_u \left(\sin\theta \cos\theta_u \sin\phi_u - \cos\theta \sin\theta_u\right) \\ \dot{\theta} = 0 \end{cases},$$

where ϕ_u and θ_u are the gimbal control angles, and $\dot{\psi}_u$ is the angular velocity of the hemisphere. Unfortunately, this strategy disables control of the robot's orientation θ, leaving only position control. The control signals for both wheels can be either identical or mirrored. The identical control signals are given by

$$\begin{cases} \phi_u = \phi_1 = \phi_2 = \arcsin\left(-\dfrac{\dot{x}_{\mathrm{ref}}}{R\dot{\psi}_u \cos\theta_u}\right) \\ \theta_u = \theta_1 = \theta_2 = \arcsin\left(-\dfrac{\dot{y}_{\mathrm{ref}}}{R\dot{\psi}_u}\right) \\ \dot{\psi}_u = \dot{\psi}_1 = \dot{\psi}_2 = \mathrm{const} \end{cases},$$

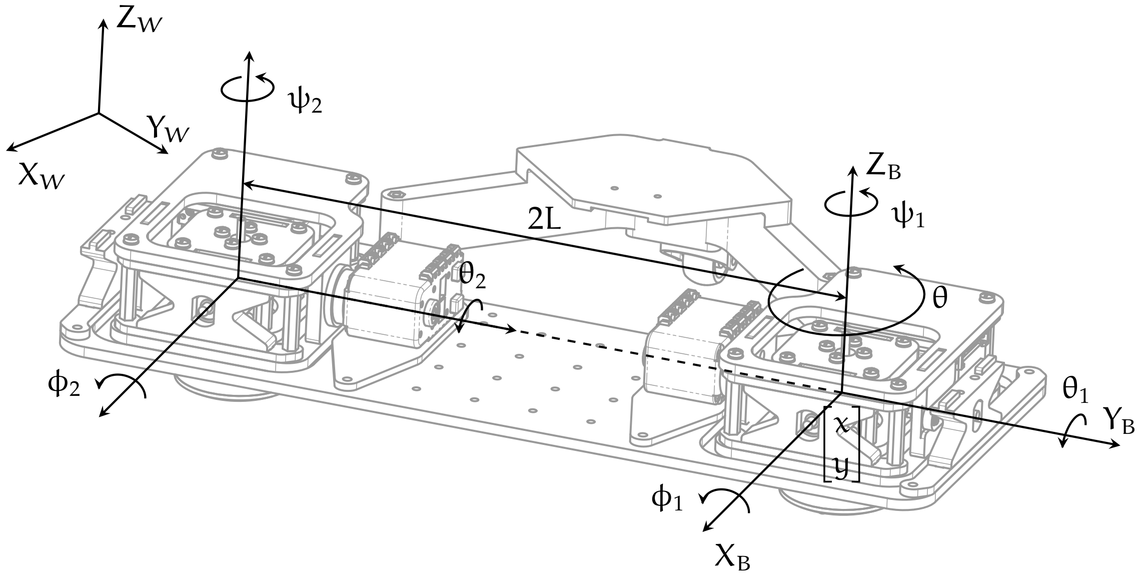


Figure 3.1: Robot reference frame and gimbal angles

while the mirrored control signals are given by

$$\begin{cases} \phi_u = \phi_1 = -\phi_2 = \arcsin\left(-\frac{\dot{x}_{\text{ref}}}{R\dot{\psi}_u \cos\theta_u}\right) \\ \theta_u = \theta_1 = -\theta_2 = \arcsin\left(-\frac{\dot{y}_{\text{ref}}}{R\dot{\psi}_u}\right) \\ \dot{\psi}_u = \dot{\psi}_1 = -\dot{\psi}_2 = \text{const} \end{cases},$$

where $\dot{x}_{\text{ref}}$ and $\dot{y}_{\text{ref}}$ are the reference velocities expressed in the robot's reference frame. The naive control law for the reference velocities was defined as

$$\begin{bmatrix} \dot{x}_{\text{ref}} \\ \dot{y}_{\text{ref}} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \left(\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix} - k \begin{bmatrix} x - x_d \\ y - y_d \end{bmatrix} \right),$$

where x_d and y_d are the components of the desired trajectory, and k is a positive constant gain.

3.2 JPTD Algorithm

The JPTD algorithm, originally introduced in [Jon17], is illustrated in Figure 3.2. It considers the configuration vector $\mathbf{q}$ defined as

$$\mathbf{q} = \begin{bmatrix} x & y & \theta & \phi_1 & \theta_1 & \psi_1 & \phi_2 & \theta_2 & \psi_2 \end{bmatrix}^T,$$

where x, y, and θ represent the robot's body position and orientation in the world reference frame $X_W Y_W Z_W$. The angles ϕ_i, θ_i, and ψ_i, $i = 1, 2$ fully describe the configuration of each HOG wheel. The robot kinematics can be modeled as a driftless system in the form

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\boldsymbol{\eta},$$

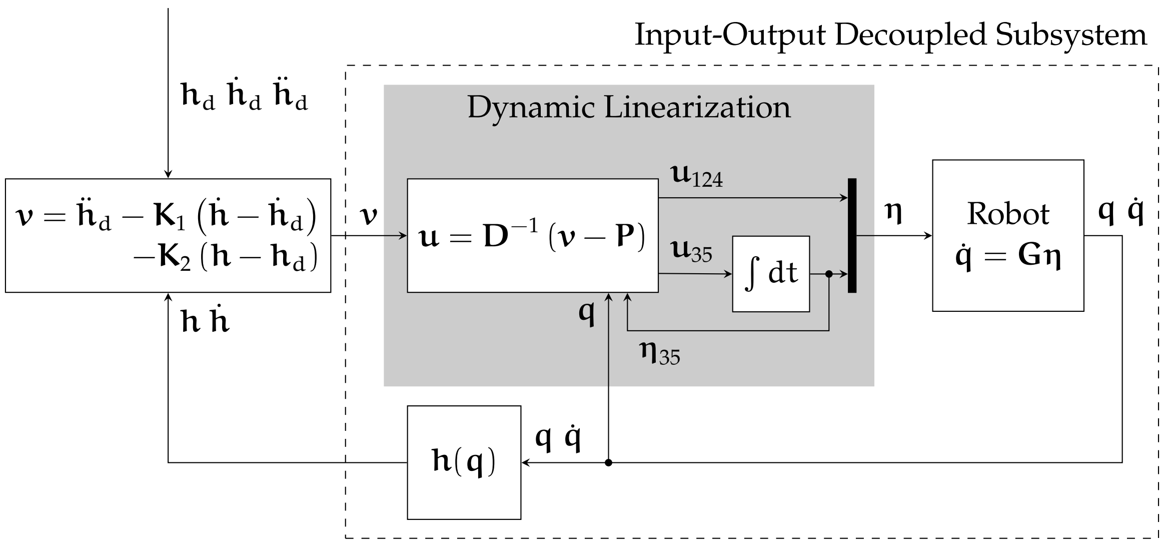


Figure 3.2: JPTD input-output decoupling with dynamic feedback linearization

where

$$\eta = \begin{bmatrix} \dot{\phi}_1 & \dot{\theta}_1 & \dot{\psi}_1 & \dot{\theta}_2 & \dot{\psi}_2 \end{bmatrix}^T,$$

and **G** is a simplified kinematic control matrix obtained in [Jon14]. The full kinematic model $\mathbf{G}_{\text{full}}$ is derived from the nonholonomic constraints and is given as

$$\mathbf{G}_{\text{full}}(q) = \begin{bmatrix} Rs_\theta & Rc_\theta c_{\phi_1} & R(s_\theta s_{\theta_1} - c_\theta s_{\phi_1} c_{\theta_1}) & 0 & 0 \\ -Rc_\theta & Rs_\theta c_{\phi_1} & -R(c_\theta s_{\theta_1} + s_\theta s_{\phi_1} c_{\theta_1}) & 0 & 0 \\ 0 & -\frac{R}{2L} c_{\phi_1} & \frac{R}{2L} s_{\phi_1} c_{\theta_1} & \frac{R}{2L} c_{\phi_2} & -\frac{R}{2L} s_{\phi_2} c_{\theta_2} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & s_{\theta_1} & 0 & -s_{\theta_2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where $s_x = \sin x$, $c_x = \cos x$, R is the radius of the hemispheres, and L is half the distance between the centers of the hemispheres. This model can be simplified by removing the influence of the hemisphere tilting angles on the robot's body movement, resulting in the reduced form

$$\mathbf{G}(q) = \begin{bmatrix} 0 & 0 & R(s_\theta s_{\theta_1} - c_\theta s_{\phi_1} c_{\theta_1}) & 0 & 0 \\ 0 & 0 & -R(c_\theta s_{\theta_1} + s_\theta s_{\phi_1} c_{\theta_1}) & 0 & 0 \\ 0 & 0 & \frac{R}{2L} s_{\phi_1} c_{\theta_1} & 0 & -\frac{R}{2L} s_{\phi_2} c_{\theta_2} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & s_{\theta_1} & 0 & -s_{\theta_2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where the highlighted elements were replaced by zeros.

Trajectory tracking is achieved using the input-output decoupling technique with dynamic feedback linearization [TM17]. The linearizing output function $\mathbf{h}$ is defined as

$$\mathbf{h}(\mathbf{q}) = \begin{bmatrix} x & y & \theta & \psi_1 & \psi_2 \end{bmatrix}^T.$$

By differentiating and factorizing this output, one can obtain the matrix $\mathbf{D}$ and the vector $\mathbf{P}$, which satisfy

$$\ddot{\mathbf{h}} = \mathbf{D}\mathbf{u} + \mathbf{P},$$

where

$$\mathbf{u} = \begin{bmatrix} \eta_1 & \eta_2 & \dot{\eta}_3 & \eta_4 & \dot{\eta}_5 \end{bmatrix}^T$$

is the internal control input with additional dynamics introduced. The problem can be reformulated as the task of controlling a double integrator system by introducing a new control signal $\boldsymbol{\nu}$

$$\boldsymbol{\nu} = \ddot{\mathbf{h}}.$$

The trajectory tracking algorithm for such a system is defined as

$$\boldsymbol{\nu} = \ddot{\mathbf{h}}_d - \mathbf{K}_1\left(\dot{\mathbf{h}} - \dot{\mathbf{h}}_d\right) - \mathbf{K}_2\left(\mathbf{h} - \mathbf{h}_d\right),$$

where $\mathbf{h}_d$ is the desired output of the system, and $\mathbf{K}_1$, $\mathbf{K}_2$ are positive definite gain matrices. To convert the linearized control signal $\boldsymbol{\nu}$ into the original input $\mathbf{u}$, the following transformation is applied

$$\mathbf{u} = \mathbf{D}^{-1}\left(\boldsymbol{\nu} - \mathbf{P}\right).$$

In summary, the following expressions were obtained during the above derivation:

$$\mathbf{D}(\mathbf{q}, \eta_{35}) = \begin{bmatrix} -R\eta_3 c_{\phi_1} c_{\theta_1} c_\theta & R\eta_3\left(c_{\theta_1} s_\theta + c_\theta s_{\phi_1} s_{\theta_1}\right) & R\left(-c_{\theta_1} c_\theta s_{\phi_1} + s_{\theta_1} s_\theta\right) & 0 & 0 \\ -R\eta_3 c_{\phi_1} c_{\theta_1} s_\theta & R\eta_3\left(-c_{\theta_1} c_\theta + s_{\phi_1} s_{\theta_1} s_\theta\right) & -R\left(c_{\theta_1} s_{\phi_1} s_\theta + c_\theta s_{\theta_1}\right) & 0 & 0 \\ \frac{R\left(\eta_3 c_{\phi_1} c_{\theta_1} - \eta_5 c_{\phi_2} c_{\theta_2}\right)}{2L} & -\frac{R\eta_3 s_{\phi_1} s_{\theta_1}}{2L} & \frac{R c_{\theta_1} s_{\phi_1}}{2L} & \frac{R\eta_5 s_{\phi_2} s_{\theta_2}}{2L} & -\frac{R c_{\theta_2} s_{\phi_2}}{2L} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{P}(\mathbf{q}, \eta_{35}) = \begin{bmatrix} \frac{R^2\eta_3\left(\eta_3 c_{\theta_1}^2 s_{\phi_1}^2 s_\theta + \eta_3 c_{\theta_1} c_\theta s_{\phi_1} s_{\theta_1} - \eta_5 c_{\theta_1} c_{\theta_2} s_{\phi_1} s_{\phi_2} s_\theta - \eta_5 c_{\theta_2} c_\theta s_{\phi_2} s_{\theta_1}\right)}{2L} \\ \frac{R^2\eta_3\left(-\eta_3 c_{\theta_1}^2 c_\theta s_{\phi_1}^2 + \eta_3 c_{\theta_1} s_{\phi_1} s_{\theta_1} s_\theta + \eta_5 c_{\theta_1} c_{\theta_2} c_\theta s_{\phi_1} s_{\phi_2} - \eta_5 c_{\theta_2} s_{\phi_2} s_{\theta_1} s_\theta\right)}{2L} \\ \frac{R\eta_5 c_{\phi_2} c_{\theta_2}\left(-\eta_3 s_{\theta_1} + \eta_5 s_{\theta_2}\right)}{2L} \\ 0 \\ 0 \end{bmatrix}.$$

Analysis of the determinant of the matrix $\mathbf{D}$

$$\det \mathbf{D} = -\frac{R^3\eta_3^2\eta_5 \cos\phi_1 \cos^2\theta_1 \sin\phi_2 \sin\theta_2}{2L}$$

shows that singularities arise when any HOG wheel halts, or when the right HOG wheel moves purely along one axis of the robot's reference frame $X_B Y_B Z_B$, or when the right HOG wheel is set to vertical. For this reason, special constraints need to be applied to the desired trajectory $\mathbf{h}_d$ to avoid these singularities.

3.3 JPTD–1D Algorithm

As an alternative to the JPTD algorithm, an even more simplified method was proposed. The two HOG wheel robot can be reduced to a (2,0) robot by considering $\theta_1 = 0$, $\theta_2 = 0$, and $\psi_1 = -\psi_2 = \psi$ which gives the configuration vector $\mathbf{q}$ as

$$\mathbf{q} = \begin{bmatrix} x & y & \theta & \phi_1 & \phi_2 & \psi \end{bmatrix}^T .$$

Then, the driftless system can be rewritten using

$$\eta = \begin{bmatrix} \dot{\phi}_1 & \dot{\phi}_2 & \dot{\psi} \end{bmatrix}^T ,$$

and

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} 0 & 0 & -R\cos\theta\sin\phi_1 \\ 0 & 0 & -R\sin\theta\sin\phi_1 \\ 0 & 0 & \frac{R}{2L}\sin\phi_1 + \frac{R}{2L}\sin\phi_2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

The linearizing output function $\mathbf{h}$ was selected as

$$\mathbf{h}(\mathbf{q}) = \begin{bmatrix} x + d\cos\theta \\ y + d\sin\theta \\ \psi \end{bmatrix} ,$$

where d is length of the "virtual shaft". It should be noted that, since the robot cannot move sideways in this method, the virtual shaft represents a point in front of the robot with which the algorithm is able to track the trajectory. Additionally, due to the smaller number of control inputs, trajectory dimensionality is reduced, and direct control over θ is lost.

Input-output decoupling is obtained in the same way as in the JPTD algorithm, but $\mathbf{u}$ is redefined as

$$\mathbf{u} = \begin{bmatrix} \eta_1 & \eta_2 & \dot{\eta}_3 \end{bmatrix}^T .$$

In summary, the following expressions were obtained during the above derivation:

$$\mathbf{D}(\mathbf{q}, \eta_3) = \begin{bmatrix} -\frac{R\eta_3 c_{\phi_1}(2Lc_\theta + ds_\theta)}{2L} & -\frac{R\eta_3 c_{\phi_2} ds_\theta}{2L} & \frac{R\left(-2Lc_\theta s_{\phi_1} - ds_\theta\left(s_{\phi_1} + s_{\phi_2}\right)\right)}{2L} \\ \frac{R\eta_3 c_{\phi_1}(-2Ls_\theta + c_\theta d)}{2L} & \frac{R\eta_3 c_{\phi_2} c_\theta d}{2L} & \frac{R\left(-2Ls_{\phi_1} s_\theta + c_\theta d\left(s_{\phi_1} + s_{\phi_2}\right)\right)}{2L} \\ 0 & 0 & 1 \end{bmatrix} ,$$

$$\mathbf{P}(\mathbf{q}, \eta_3) = \begin{bmatrix} \frac{R^2\eta_3^2\left(2Ls_{\phi_1}^2 s_\theta + 2Ls_{\phi_1} s_{\phi_2} s_\theta - c_\theta ds_{\phi_1}^2 - 2c_\theta ds_{\phi_1} s_{\phi_2} - c_\theta ds_{\phi_2}^2\right)}{4L^2} \\ -\frac{R^2\eta_3^2\left(2Lc_\theta s_{\phi_1}^2 + 2Lc_\theta s_{\phi_1} s_{\phi_2} + ds_{\phi_1}^2 s_\theta + 2ds_{\phi_1} s_{\phi_2} s_\theta + ds_{\phi_2}^2 s_\theta\right)}{4L^2} \\ 0 \end{bmatrix} .$$

Analysis of the determinant of the matrix $\mathbf{D}$

$$\det \mathbf{D} = -\frac{dR^2\eta_3^2 \cos\phi_1 \cos\phi_2}{2L}$$

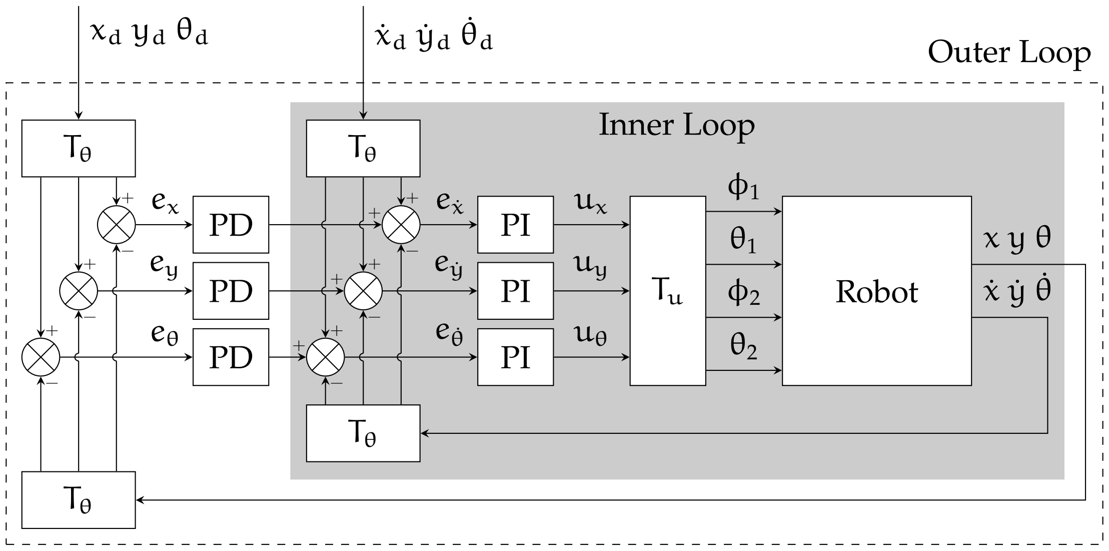


Figure 3.3: Structure of the cascade PID control

shows that singularities occur only when any HOG wheel stops rotating or the virtual shaft has no length. The length of the virtual shaft directly corresponds to the offset between the actual and desired positions during trajectory tracking. To reduce this offset, the desired trajectory $\mathbf{h}_d$ can be constructed as

$$\mathbf{h}_d = \begin{bmatrix} x_d + d\cos\theta_{dir} \\ y_d + d\sin\theta_{dir} \\ \psi_d \end{bmatrix},$$

where $\theta_{dir} = \mathrm{atan2}(\dot{y}_d, \dot{x}_d)$.

3.4 Cascade PID Control

For comparison, a basic control algorithm was proposed. Cascade PID control is a widely used technique in industry for practical control tasks [ÅH95]. By skillfully transforming the problem, it becomes possible to apply well-known and simple methods for designing and tuning PID controllers. This section describes both procedures.

3.4.1 Algorithm

The cascade PID control, shown in Figure 3.3, consists of two feedback loops. Reference frames and symbols are identical to those used in the JPTD algorithm. In the robot's reference frame $X_BY_BZ_B$, controlling each of the x, y, and θ movements is independent, so three separate controllers can be introduced for the task of tracking reference trajectories. To achieve this, the transformation matrix $\mathbf{T}_\theta$ rotates error vectors expressed in the world reference frame $X_WY_WZ_W$ to the robot's frame $X_BY_BZ_B$

$$\mathbf{T}_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

For practical reasons, instead of transforming the errors directly, the transformation T_θ is applied to the reference values and the process values separately.

The inner feedback loop controls the velocities of the robot using the gimbal angles ϕ_1, θ_1, ϕ_2, and θ_2. The trajectory reference velocities are applied as setpoints to the PI controllers. The transformation $\mathbf{T}_u$, often called the "motor mixer" matrix, is derived from the robot's kinematics, expressed in the robot's reference frame in the following form

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix} -R\dot{\psi}_1 \sin\phi_1 \cos\theta_1 \\ -R\dot{\psi}_1 \sin\theta_1 \\ \frac{R}{2L}\dot{\psi}_1 \sin\phi_1\cos\theta_1 - \frac{R}{2L}\dot{\psi}_2 \sin\phi_2 \cos\theta_2 \end{bmatrix}.$$

After substituting $\dot{\psi}_1 = -\dot{\psi}_2 = \dot{\psi} = \text{const}$ and applying the small-angle approximation

$$\begin{cases}\sin\alpha \approx \alpha \\ \cos\alpha \approx 1\end{cases},$$

we obtain

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix} -R\dot{\psi} & 0 & 0 \\ 0 & -R\dot{\psi} & 0 \\ \frac{R\dot{\psi}}{2L} & 0 & \frac{R\dot{\psi}}{2L} \end{bmatrix} \begin{bmatrix}\phi_1\\ \theta_1\\ \phi_2\end{bmatrix},$$

and finally

$$\begin{bmatrix}\phi_1\\ \theta_1\\ \phi_2\end{bmatrix} = \begin{bmatrix} -\frac{1}{R\dot{\psi}} & 0 & 0 \\ 0 & -\frac{1}{R\dot{\psi}} & 0 \\ \frac{1}{R\dot{\psi}} & 0 & \frac{2L}{R\dot{\psi}} \end{bmatrix} \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}.$$

The gimbal angles θ_1 and θ_2 control movement along the Y_B axis. To ensure motion without slippage, both angles must satisfy the condition $\theta_2 = -\theta_1$. In summary, the three controller outputs u_x, u_y, and u_θ are merged using the formula

$$\begin{bmatrix}\phi_1\\ \theta_1\\ \phi_2\\ \theta_2\end{bmatrix} = \begin{bmatrix} -\frac{1}{R\dot{\psi}} & 0 & 0 \\ 0 & -\frac{1}{R\dot{\psi}} & 0 \\ \frac{1}{R\dot{\psi}} & 0 & \frac{2L}{R\dot{\psi}} \\ 0 & \frac{1}{R\dot{\psi}} & 0 \end{bmatrix} \begin{bmatrix}u_x\\ u_y\\ u_\theta\end{bmatrix} = \mathbf{T}_u\mathbf{u}.$$

The outer feedback loop (see Figure 3.3) controls the position of the robot by applying corrective reference velocities to the inner loop. This allows the robot's position to converge to the reference trajectory.

3.4.2 Inner Loop Tunning

From the inner loop's perspective, the system can be modeled as an inertial system with lag, as described in Section 2.3.3. Similarly, the model parameters K, T, and L can be obtained by applying a step input to each of the u_x, u_y, and u_θ signals separately. To tune the internal PI controllers, the Chien-Hrones-Reswick method with

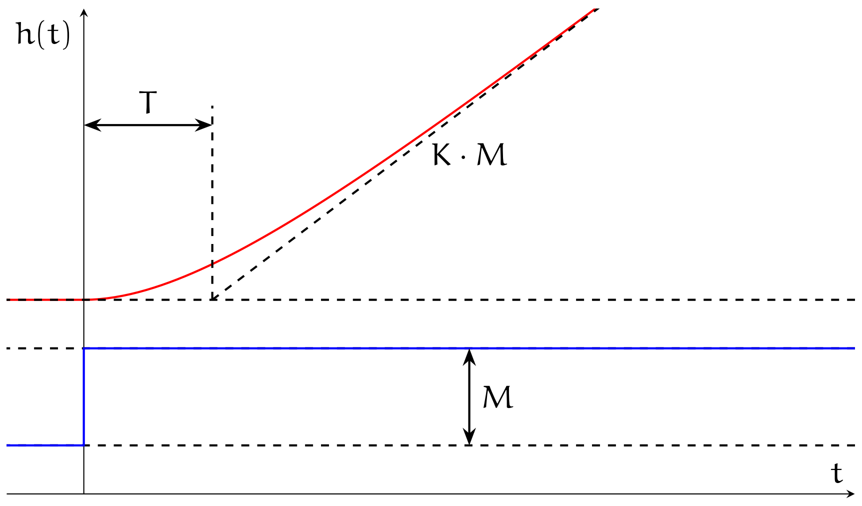


Figure 3.4: Geometric properties of the step response of an integrating system

0% overshoot [ÅH95] is used, according to the formulas

$$\begin{cases} K_p = 0.35\dfrac{T}{KL} \\ K_i = 0.29\dfrac{1}{KL} \\ K_d = 0 \end{cases} .$$

The results of the tuning procedure are provided in Appendix C. Using the transformation T_u, the model gain K should ideally equal 1. If $K \approx 1$, it can be assumed that the step response was collected correctly.

3.4.3 Outer Loop Tunning

From the outer loop's perspective, the system, with internal controllers running, can be modeled as an integrating system with inertia. To tune the PID controller for such a system, the λ-tuning method [ÅH95] was used. The method, shown in Figure 3.4, involves drawing additional straight lines on the step response (red plot) of an unknown integrating system to represent:

- the signal level at steady state before applying the step,
- the tangent line to the pure integrating portion of the response.

Analysis of the line's intersection and slope reveals the system's key geometric properties, namely, the gain K, and time constant T, which fully characterize the plant. Because the outer loop controls position by setting the reference velocity of the system, again one should expect that K equals 1 under ideal conditions. If $K \approx 1$, it can be assumed that the step response was collected correctly.

The λ-tuning method consists of calculating the ideal controller $C(s)$ using the plant transfer function $P(s)$, and the desired closed-loop response transfer function $G_z(s)$.

The desired step response was selected as

$$G_z(s) = \frac{1}{\lambda T s + 1},$$

where λ is a user defined parameter related to the time constant of the closed-loop system. The plant was identified as

$$P(s) = \frac{K}{s\,(Ts + 1)}.$$

Using the standard closed-loop transfer function of a system with error feedback

$$G_z(s) = \frac{C(s)P(s)}{1 + C(s)P(s)},$$

one can solve for $C(s)$ using the formula

$$C(s) = \frac{1}{P(s)}\frac{G_z(s)}{1 - G_z(s)}.$$

Finally, by grouping the coefficients of $\frac{1}{s}$, which defines K_i, and s, which defines K_d, the formulas for the ideal PID controller take the form

$$\begin{cases} K_p = \dfrac{1}{\lambda K T} \\ K_i = 0 \\ K_d = \dfrac{1}{\lambda K} \end{cases}.$$

Chapter 4

Experiments

This chapter presents the results of the experiments conducted as a part of the thesis. The robot's geometric parameters were kept constant during all experiments, with values $L = 0.13$ m and $R = 0.05$ m. The experiments were carried out on a flat stone floor without noticeable grouting, as shown in Figure 4.1. The use of an optical sensor required selecting a floor surface with sufficiently nonuniform coloring to ensure reliable sensor readings. Furthermore, due to the use of the BEMF algorithm for BLDC motor control, all implemented control algorithms assume constant motor speeds. Motor control is handled by the internal PID controller, rather than relying on the control signals computed by the tested algorithms. For algorithms in which the control signals involve regulating servo velocity, additional integrators were introduced to compute the corresponding angle setpoints.

The reference trajectory was a variation of the Lissajous curve, as used in [Jon17], and is defined by

$$\begin{cases} x_d(t) = a \sin\left(\frac{1}{2}\omega t\right) \\ y_d(t) = a \cos\left(\omega t + \frac{\pi}{4}\right) \\ \theta_d(t) = \operatorname{atan2}\Big(\dot{y}_d(t),\, \dot{x}_d(t)\Big) \end{cases},$$

where $\omega = \frac{4\pi}{T}$, $T = 30$ s is the period of one full loop, and $a = 1$ m is the trajectory span. The algorithms were also tested on various trajectory shapes, including circular, lemniscate, and linear however, the one presented above was selected for illustrative purposes. During the experiments, the trajectory tracking performance of all algorithms was evaluated for three angular velocity settings of both hemispheres: 200 $\frac{\text{rad}}{\text{s}}$, 350 $\frac{\text{rad}}{\text{s}}$, and 500 $\frac{\text{rad}}{\text{s}}$. For each test, the robot's initial position was set at the center of the room, corresponding to the origin of the world reference frame $X_W Y_W Z_W$. At the beginning of each run, the trajectory tracking algorithm and the startup procedure for the hemisphere motors were initiated simultaneously.

The plots presented in this chapter follow a consistent format. Dotted lines represent the reference values of trajectory components and actuator setpoints. Solid lines indicate the actual control signals and the corresponding states achieved by the robot.

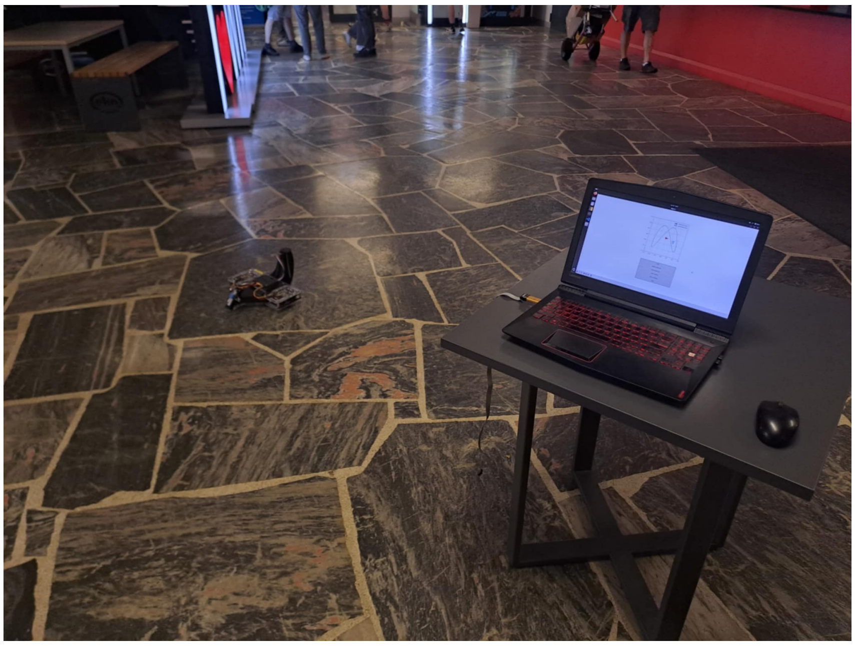


Figure 4.1: Floor in the WUST C–1 building selected for the experiments

4.1 Simultaneous Algorithm

This section presents the results of experiments with the simultaneous algorithm (described in Section 3.1). Figures 4.2–4.7 present the results for identical control inputs applied to both hemispheres, while Figures 4.8–4.13 show the results for mirrored control inputs. The control signals were clamped* to the range of $\pm 3°$.

For both variants, the results show that the robot's orientation does not remain constant, in contrast to the simulation results reported in [Jon17]. The variant with identical control signals causes the robot to spin, as shown in Figure 4.14. On the other hand, mirrored control inputs do not significantly influence the robot's angular velocity, since frictional torques cancel each other out, leaving only minor variations due to floor and hemisphere imperfections. The naive control law was unable to track the trajectory satisfactorily. In terms of planar movement, the robot's behavior was consistent with the results obtained in the previously conducted simulations [Jon17]. The tracking errors e_x and e_y also oscillated around zero, though with greater amplitude.

4.2 JPTD Algorithm

Next, the JPTD algorithm was applied (see Section 3.2). The results of its tests are shown in Figures 4.15–4.20. The algorithm requires special consideration regarding the reference trajectory. In this case, the reference orientation $\theta_d(t)$ was redefined as

$$\theta_d(t) = \operatorname{atan2}\Big(\dot{y}_d(t),\ \dot{x}_d(t)\Big) + \frac{\pi}{4},$$

*At low hemisphere speeds, the BLDC motor controllers were unable to maintain motor operation under high load conditions, which arise when the gimbals are tilted excessively. For each of the examined algorithms, similar constraints were imposed, with their values determined experimentally.

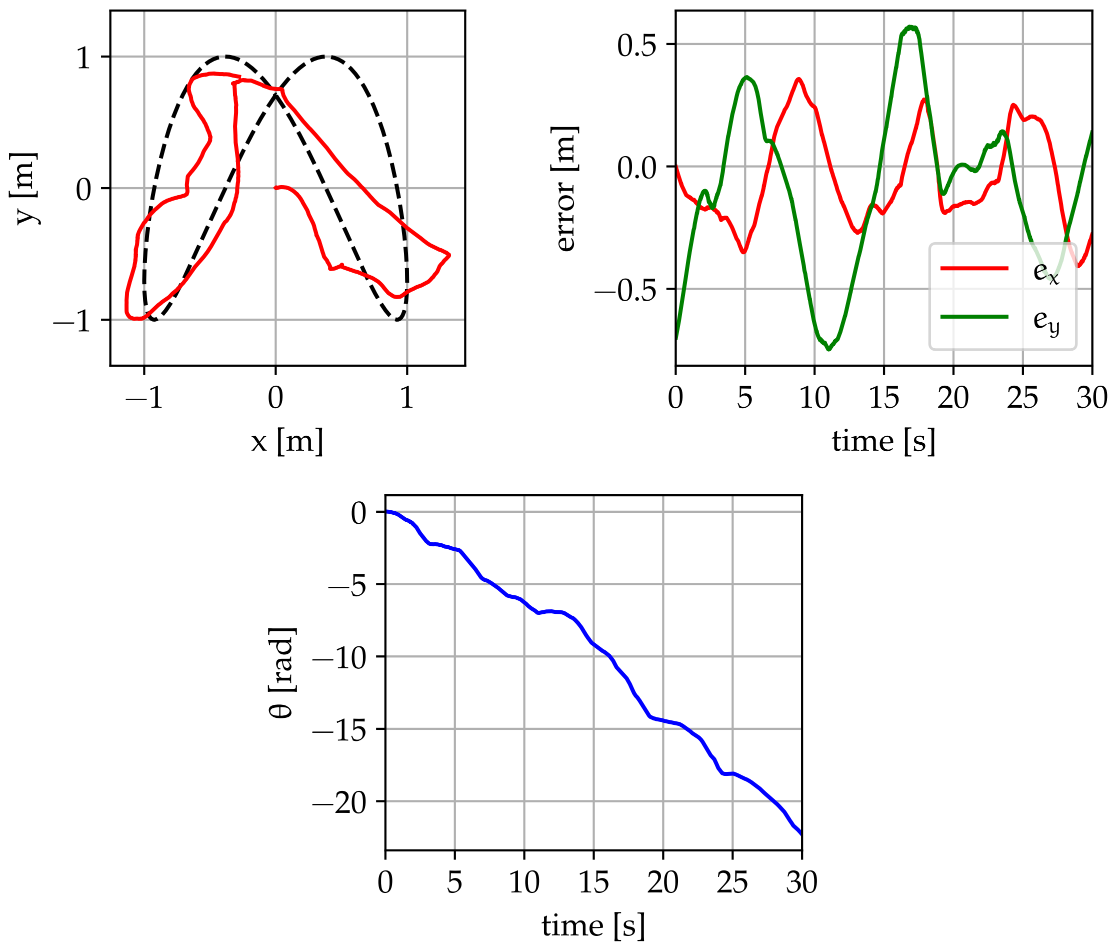


Figure 4.2: Results of trajectory tracking using simultaneous algorithm with identical controls for $\dot{\psi} = \text{-}200\ \frac{\text{rad}}{\text{s}}$, and $k = 2$

to avoid tracking trajectories aligned with the robot's reference frame axes X_B or Y_B, where the decoupling matrix becomes singular. The control signals were clamped to the range of $\pm 4°$.

The orientation tracking performance was significantly worse than the position tracking, exhibiting noticeable offset and multiple rotations for higher hemisphere velocities, not observed in simulations [Jon17]. Unfortunately, the gain matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ proposed in [Jon17] failed to ensure convergence to the desired trajectory. After increasing the gains to $\mathbf{K}_1 = \mathbf{K}_2 = 200 \cdot \mathbf{I}$, convergence in planar motion was achieved however, the robot's orientation remained unregulated. The smallest gain values that ensured convergence across all components of the desired trajectory were determined to be 1000. As a result, the integrator blocks in the control signals saturated almost instantly, leading to chattering in the angle setpoints. Assigning higher values only to the orientation-related elements of the gain matrices leads to a decrease in overall performance.

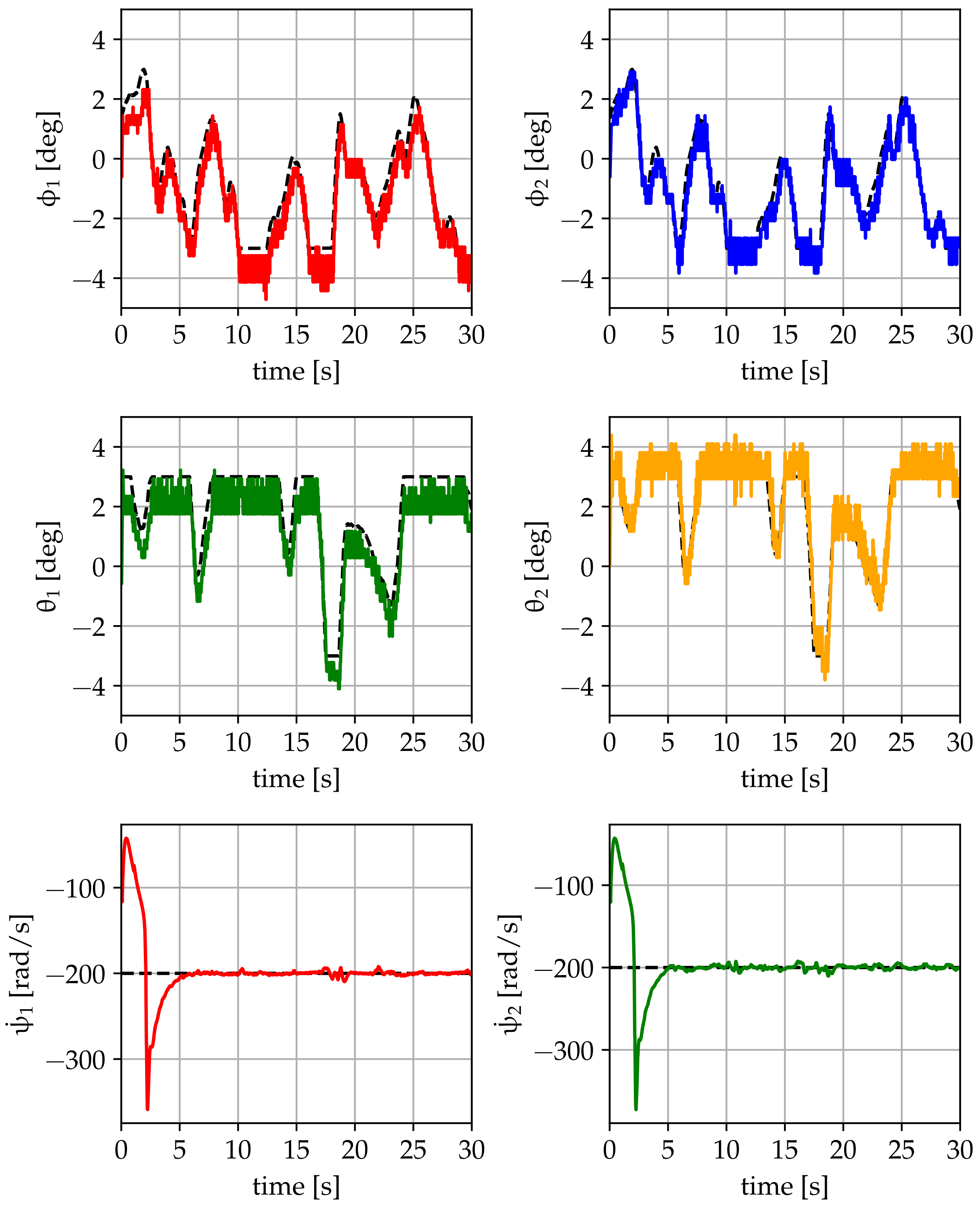


Figure 4.3: Control inputs during trajectory tracking using simultaneous algorithm with identical controls for $\dot{\psi}$ = -200 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

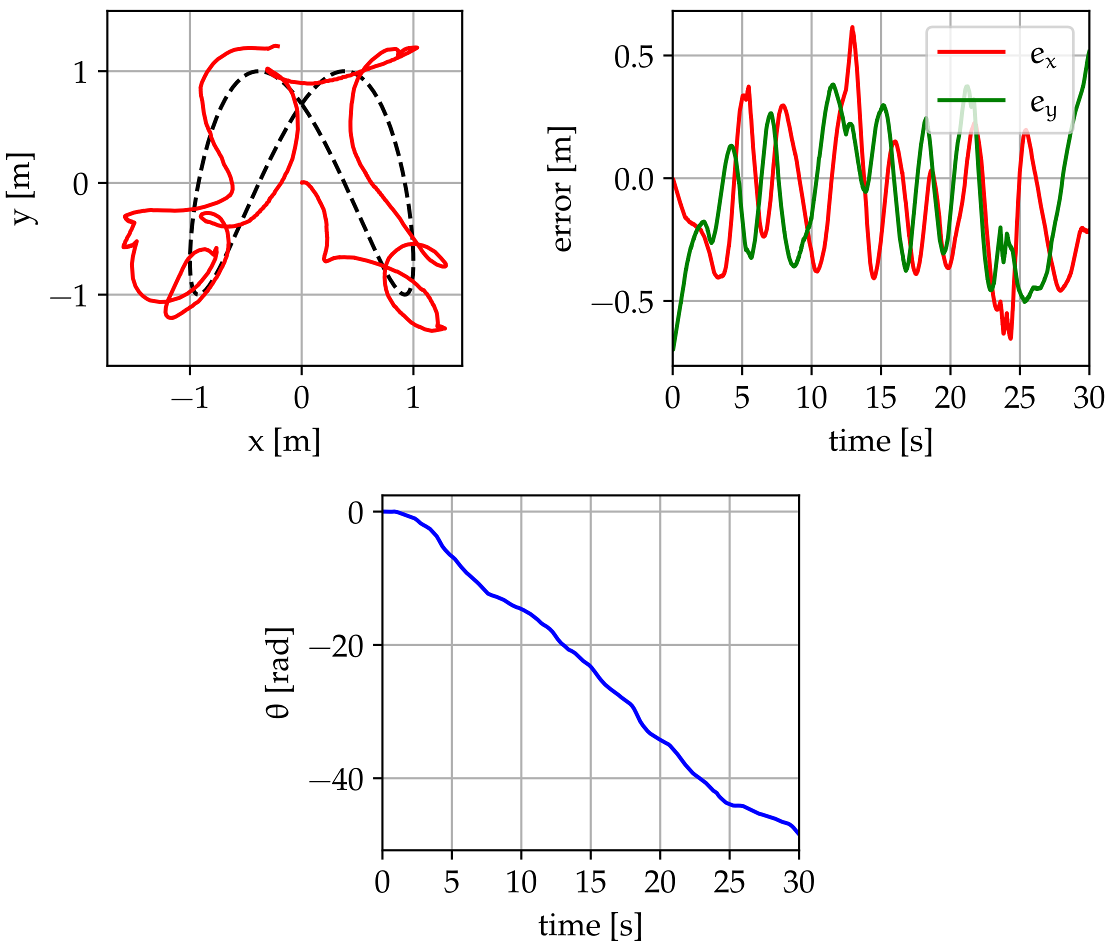


Figure 4.4: Results of trajectory tracking using simultaneous algorithm with identical controls for $\dot{\psi}$ = -350 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

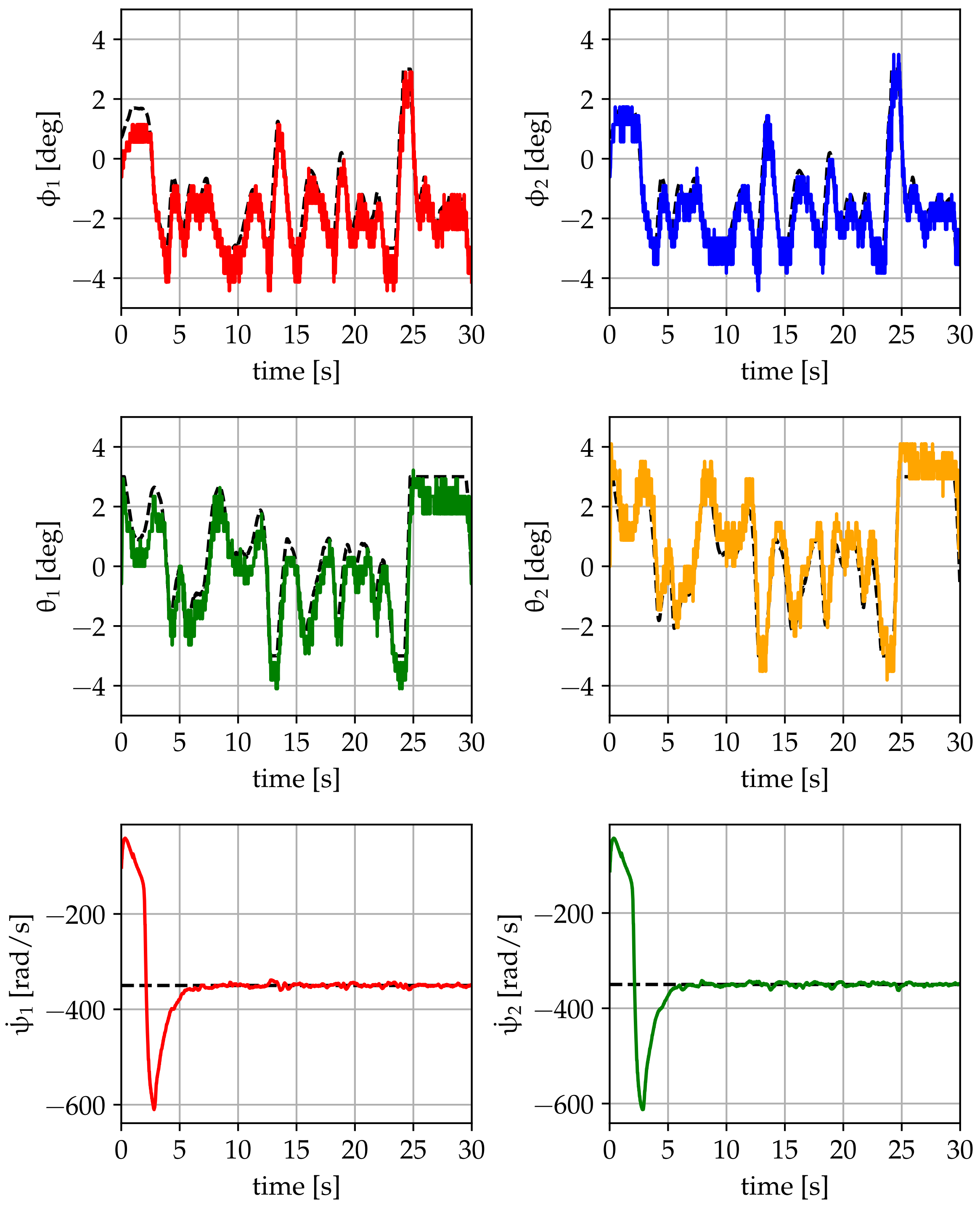


Figure 4.5: Control inputs during trajectory tracking using simultaneous algorithm with identical controls for $\dot{\psi}$ = -350 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

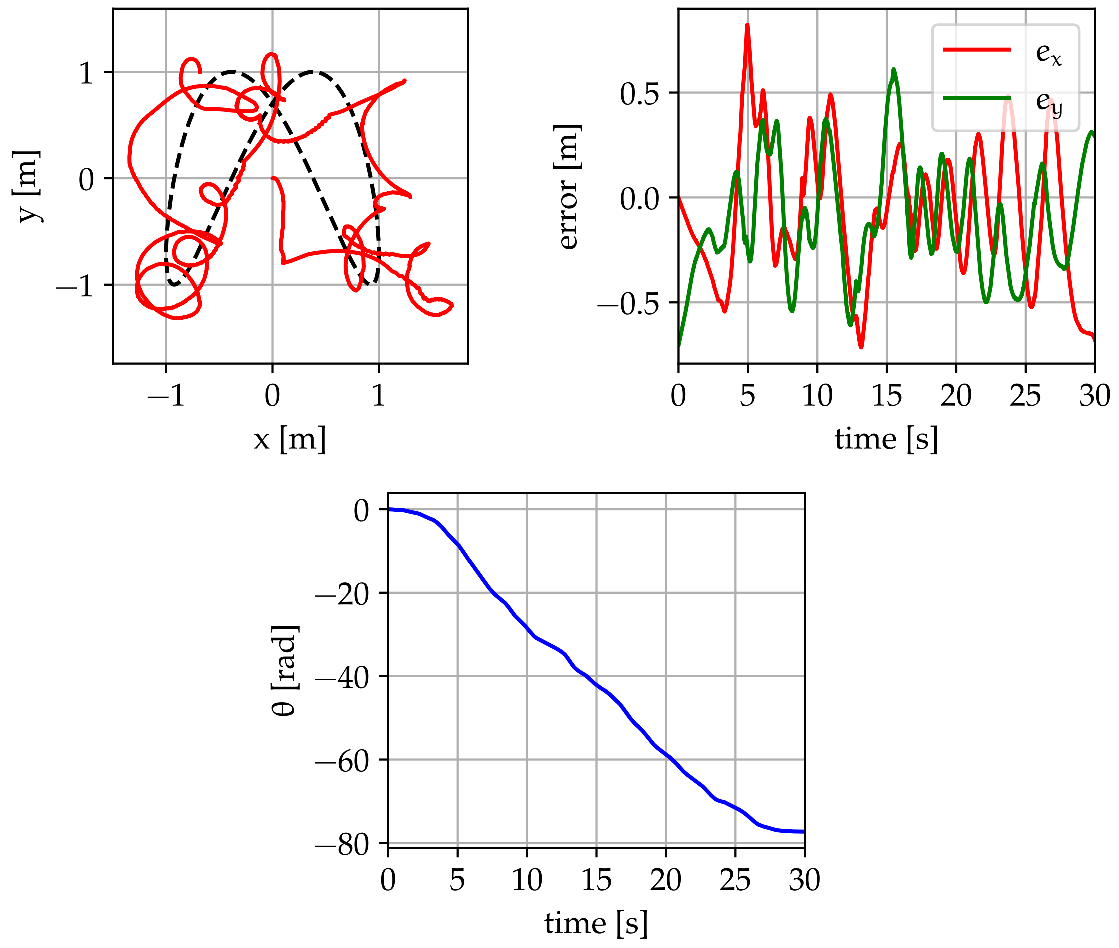


Figure 4.6: Results of trajectory tracking using simultaneous algorithm with identical controls for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

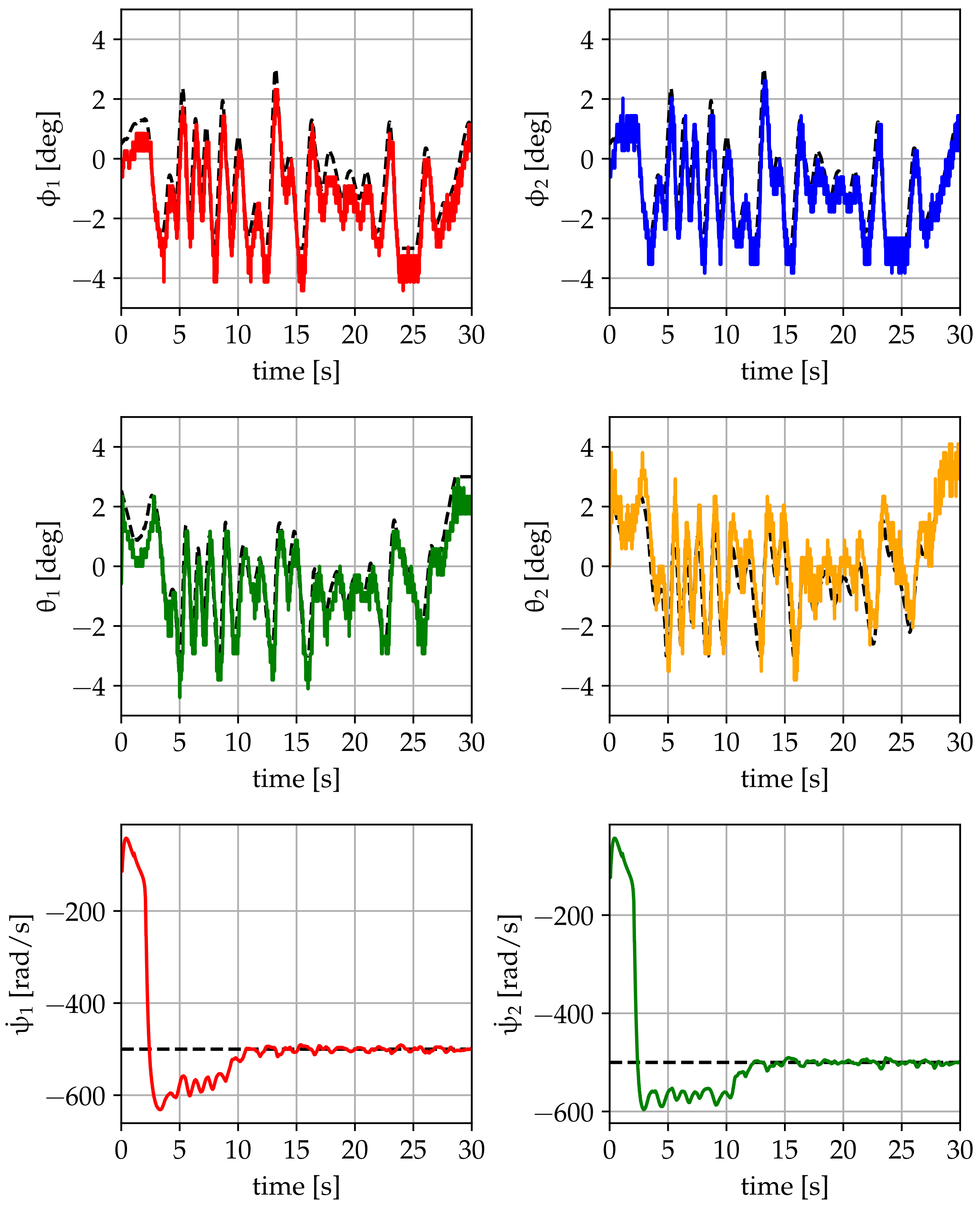


Figure 4.7: Control inputs during trajectory tracking naive simultaneous algorithm with identical controls for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

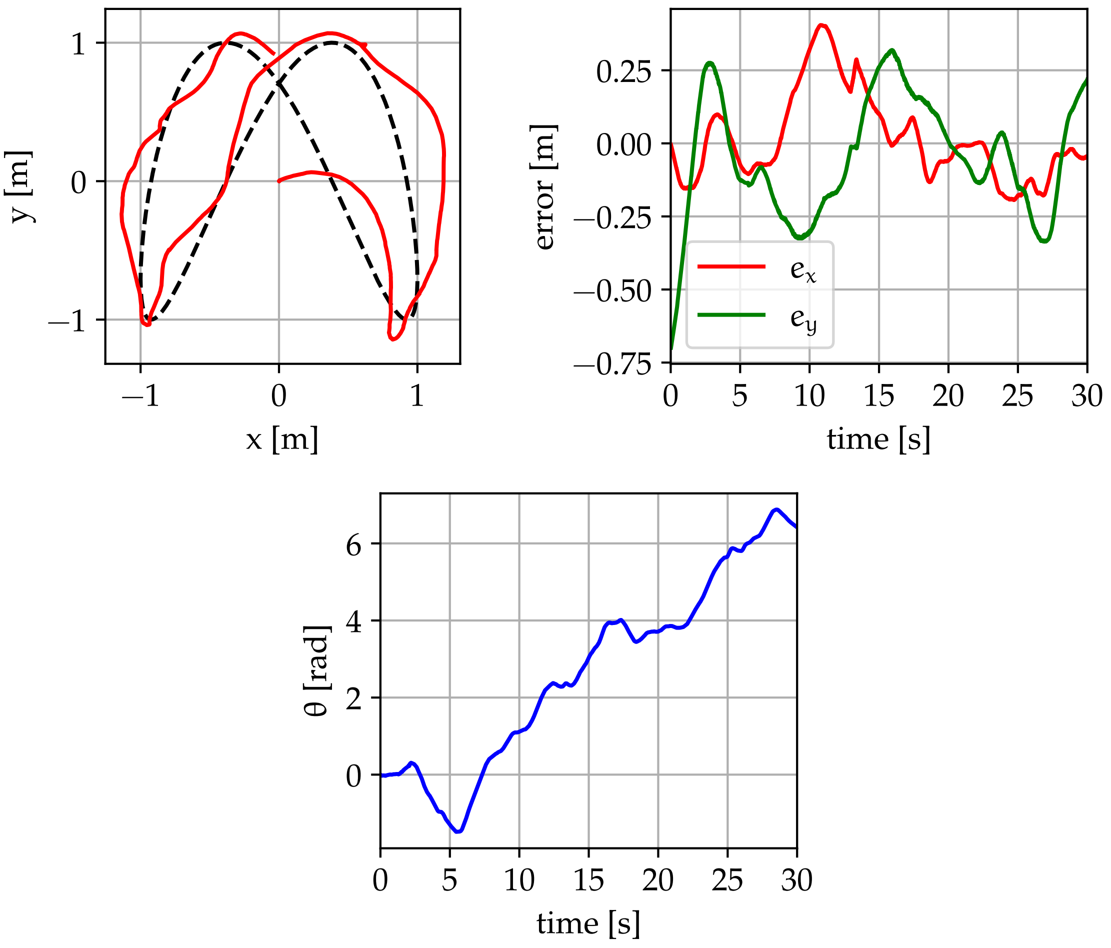


Figure 4.8: Results of trajectory tracking using simultaneous algorithm with mirrored controls for $\dot{\psi}$ = -200 $\frac{\text{rad}}{\text{s}}$, and k = 2

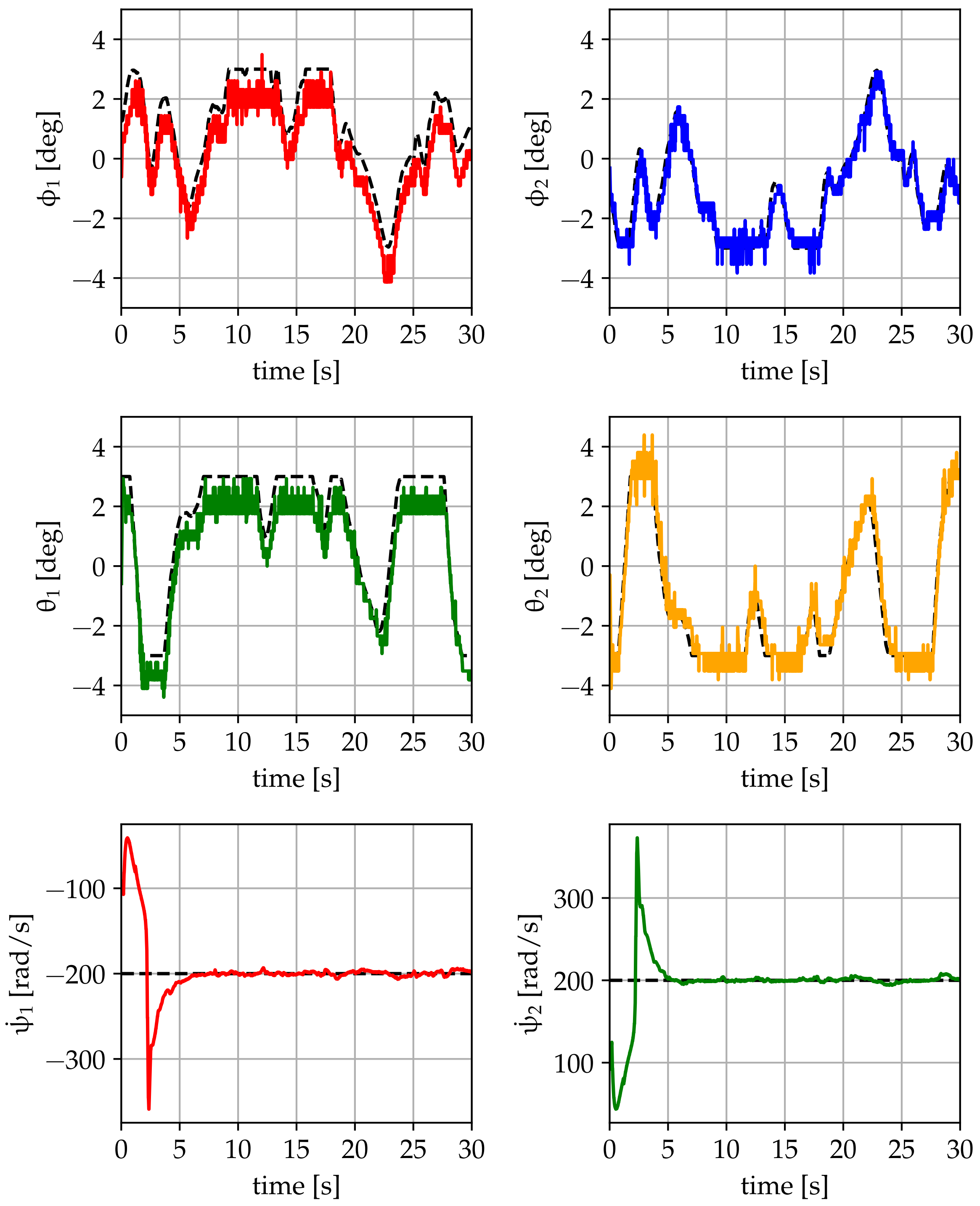


Figure 4.9: Control inputs during trajectory tracking using simultaneous algorithm with mirrored controls for $\dot{\psi}$ = -200 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

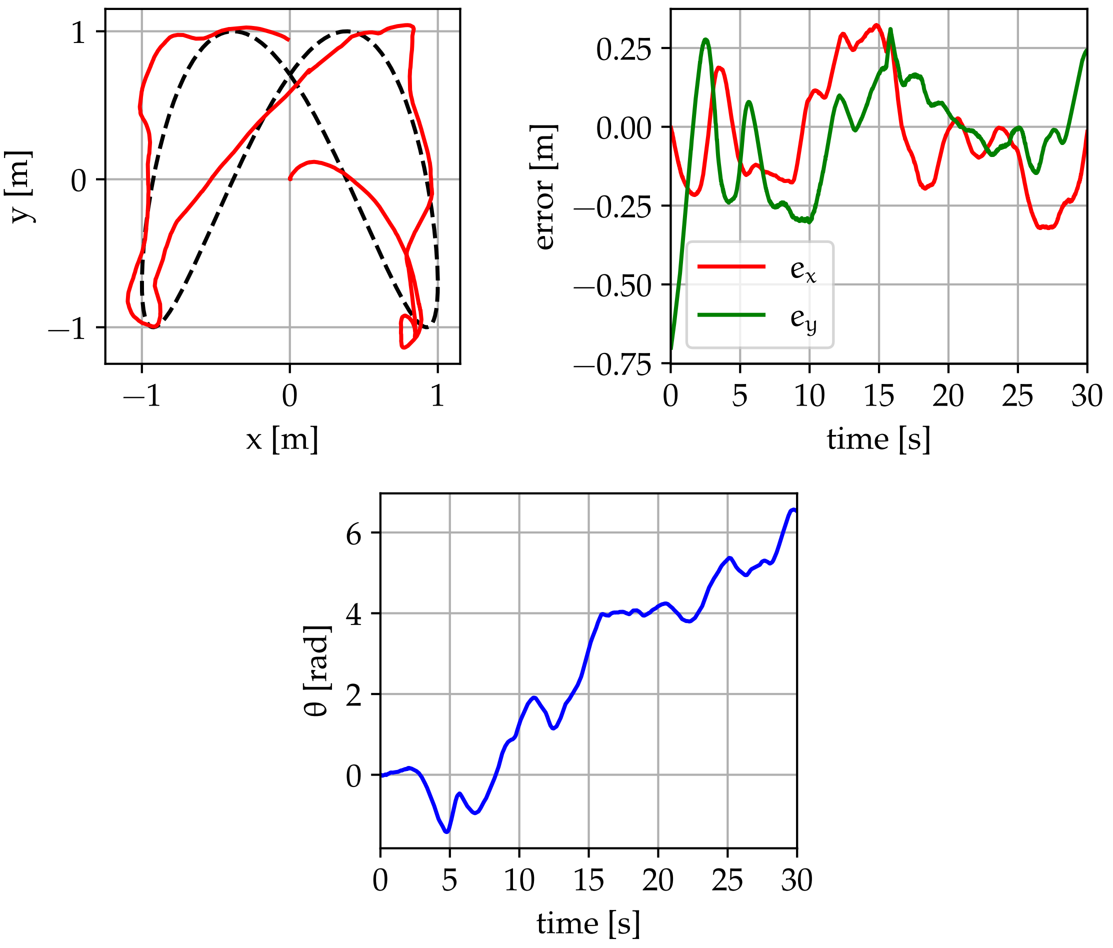


Figure 4.10: Results of trajectory tracking using simultaneous algorithm with mirrored controls for $\dot{\psi}$ = -350 $\frac{\text{rad}}{\text{s}}$, and k = 2

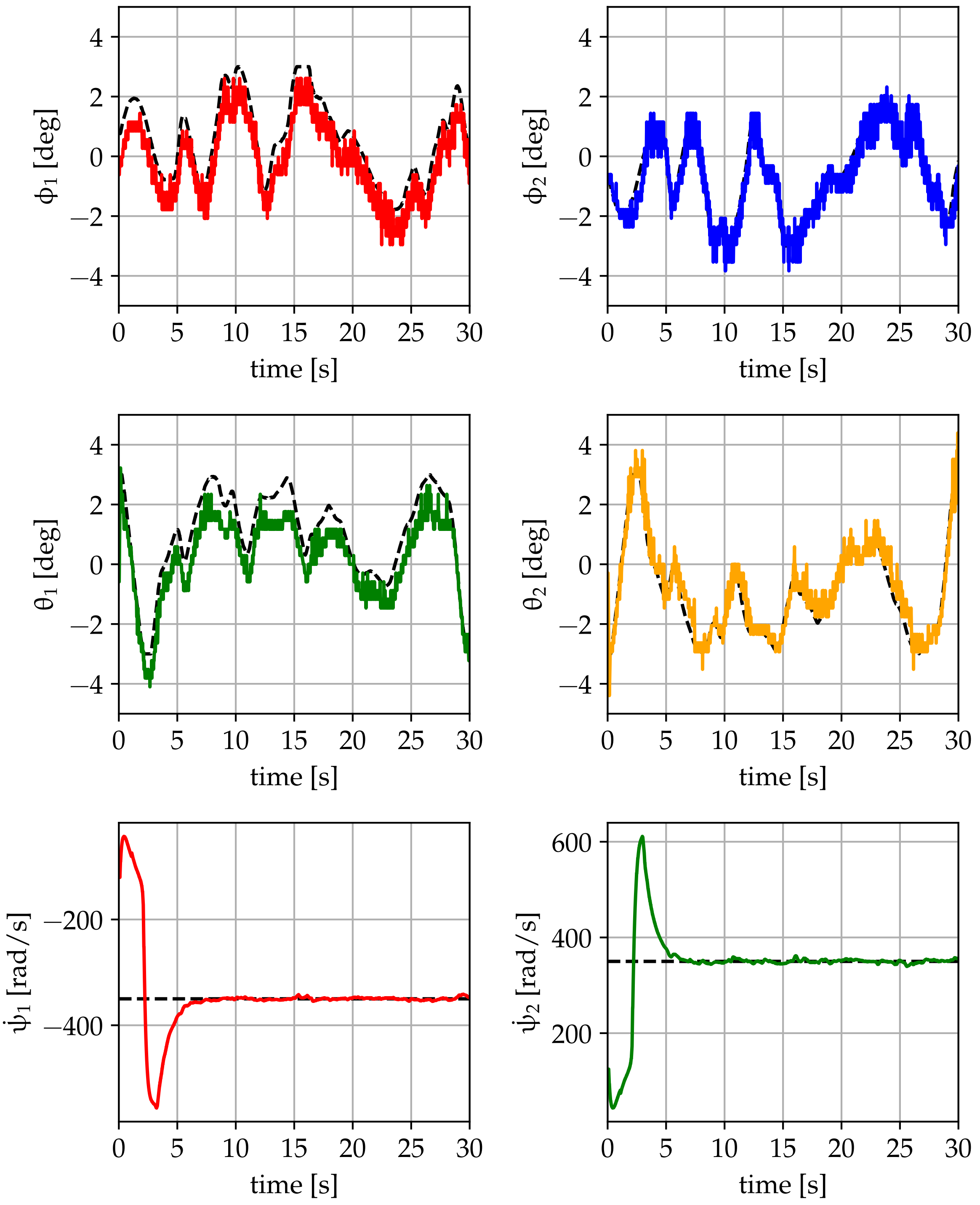


Figure 4.11: Control inputs during trajectory tracking using simultaneous algorithm with mirrored controls for $\dot{\psi}$ = -350 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

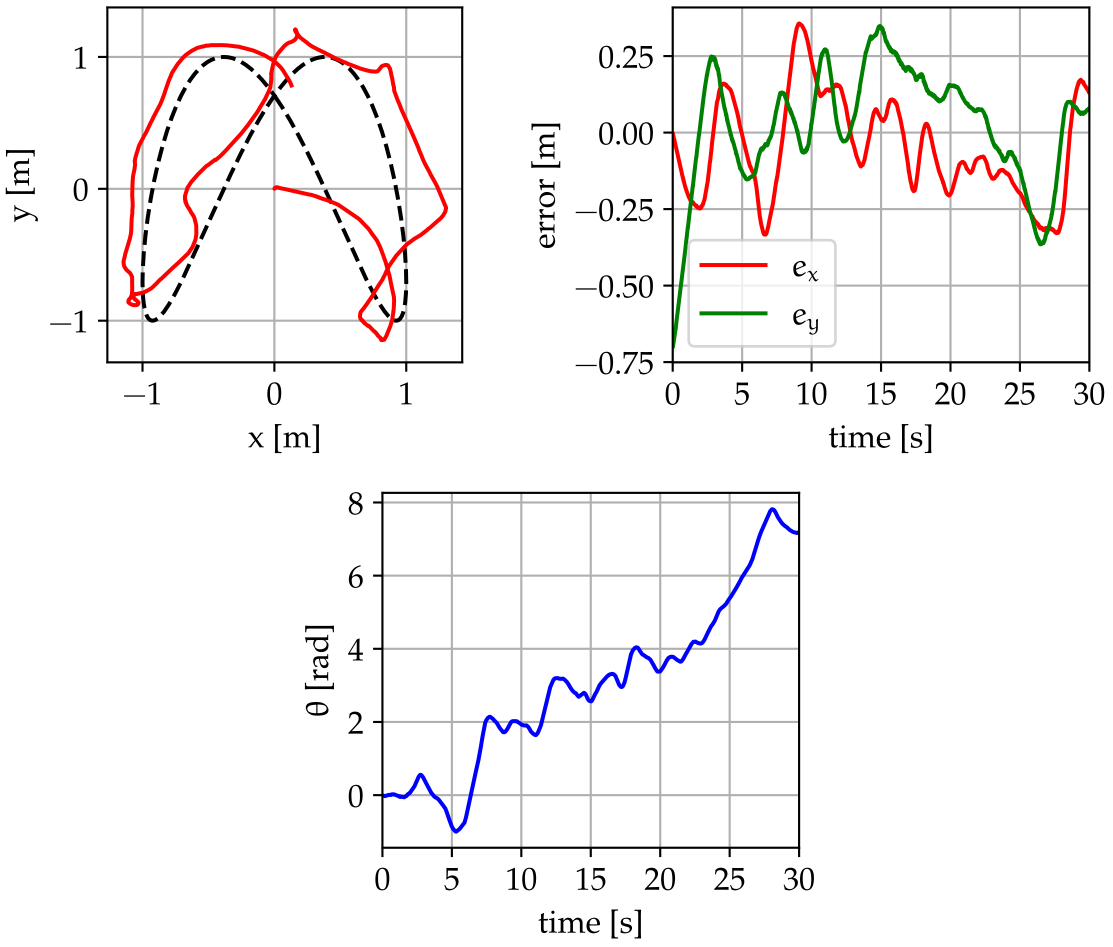


Figure 4.12: Results of trajectory tracking using simultaneous algorithm with mirrored controls for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

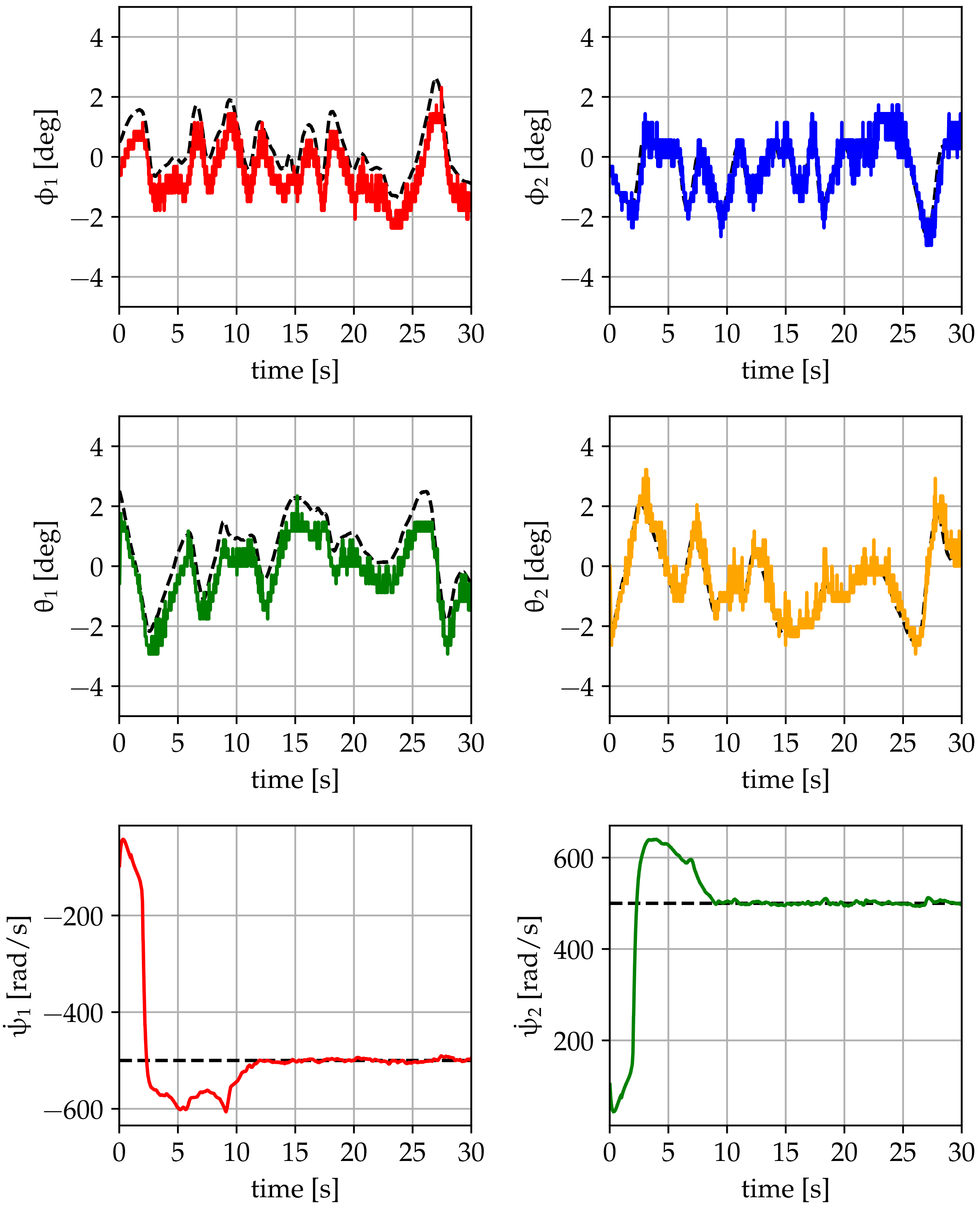


Figure 4.13: Control inputs during trajectory tracking using simultaneous algorithm with mirrored controls for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$, and $k = 2$

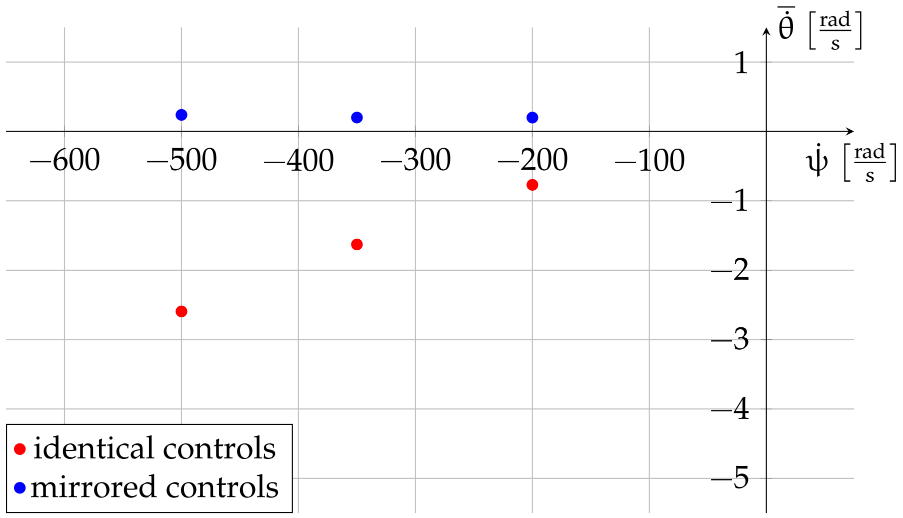


Figure 4.14: Mean robot's angular velocity vs. hemisphere velocities while using simultaneous algorithm

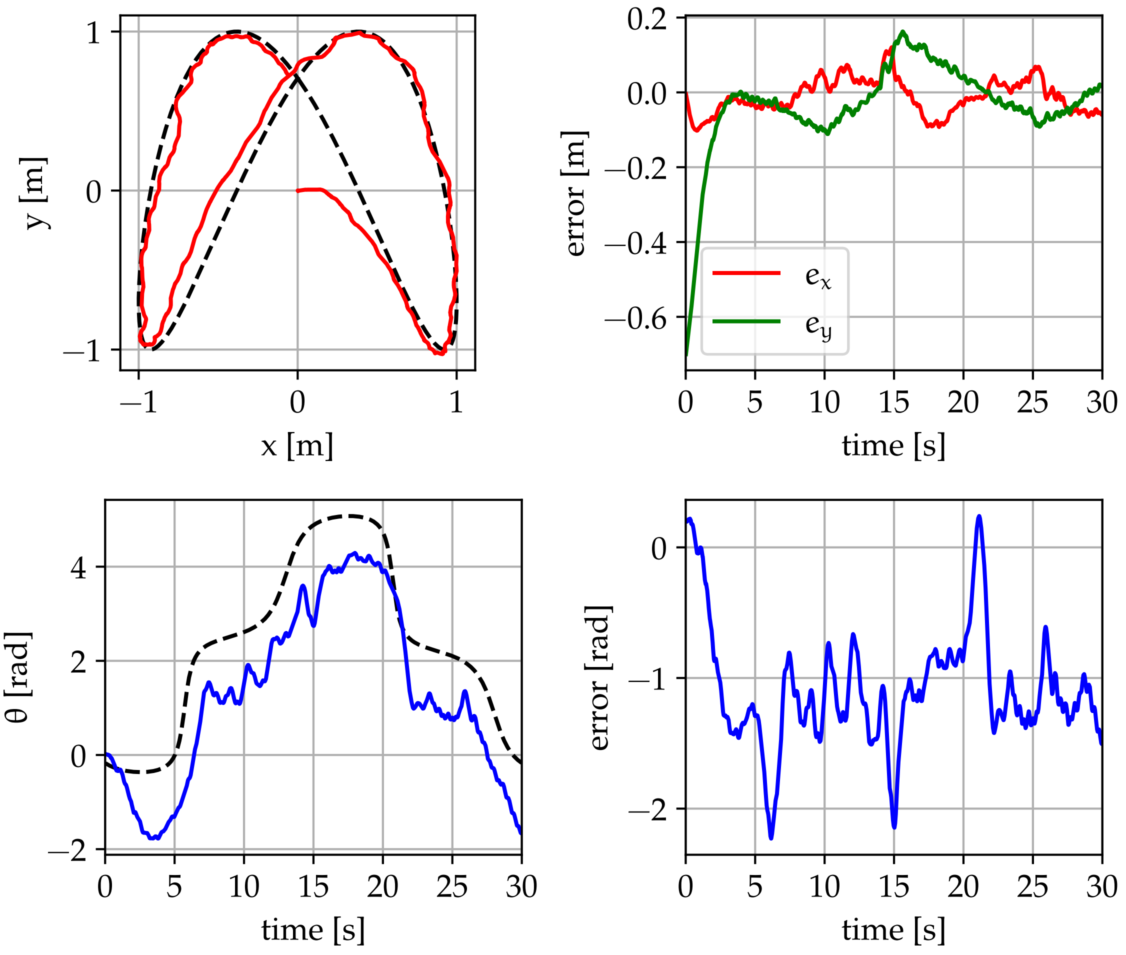


Figure 4.15: Results of trajectory tracking using JPTD algorithm for $\dot{\psi}_1 =$ -200 $\frac{\text{rad}}{\text{s}}$, $\dot{\psi}_2 = 200$ $\frac{\text{rad}}{\text{s}}$, $\mathbf{K}_1 = 1000 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 1000 \cdot \mathbf{I}$

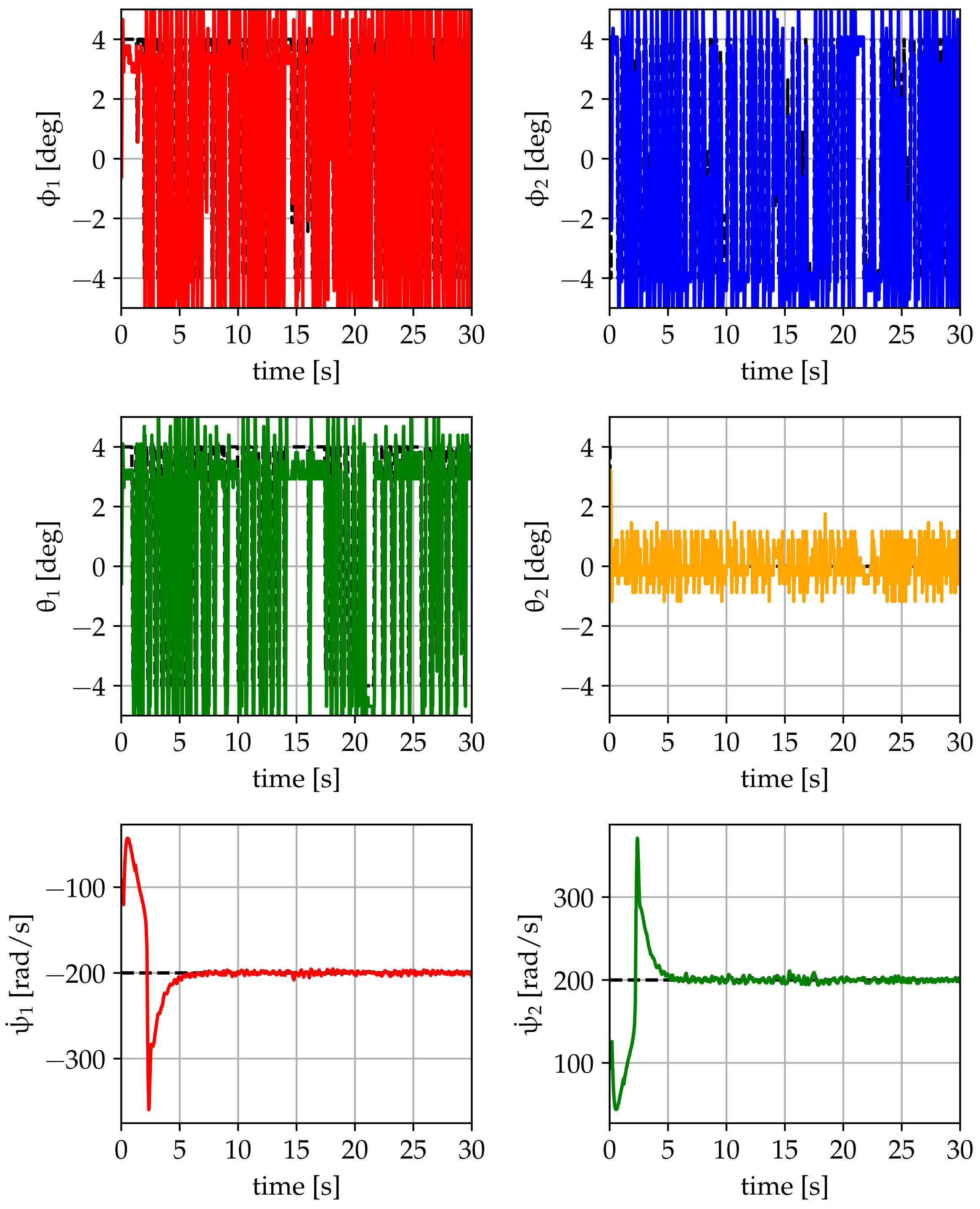


Figure 4.16: Control inputs during trajectory tracking using JPTD algorithm for $\dot{\psi}_1 =$ -200 $\frac{\text{rad}}{\text{s}}$, $\dot{\psi}_2 = 200\ \frac{\text{rad}}{\text{s}}$, $\mathbf{K}_1 = 1000 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 1000 \cdot \mathbf{I}$

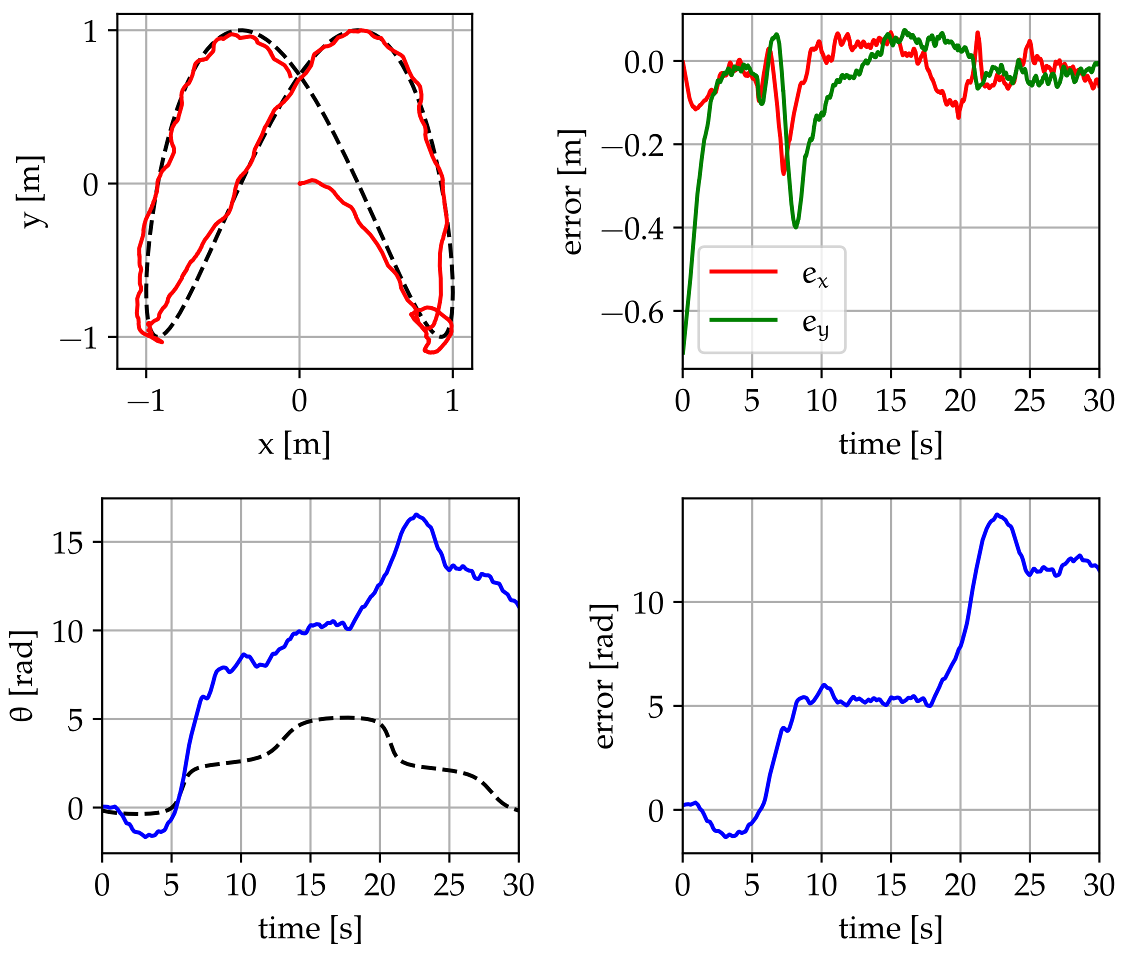


Figure 4.17: Results of trajectory tracking using JPTD algorithm for $\dot{\psi}_1$ = -350 $\frac{\text{rad}}{\text{s}}$, $\dot{\psi}_2 = 350\ \frac{\text{rad}}{\text{s}}$, $\mathbf{K}_1 = 1000 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 1000 \cdot \mathbf{I}$

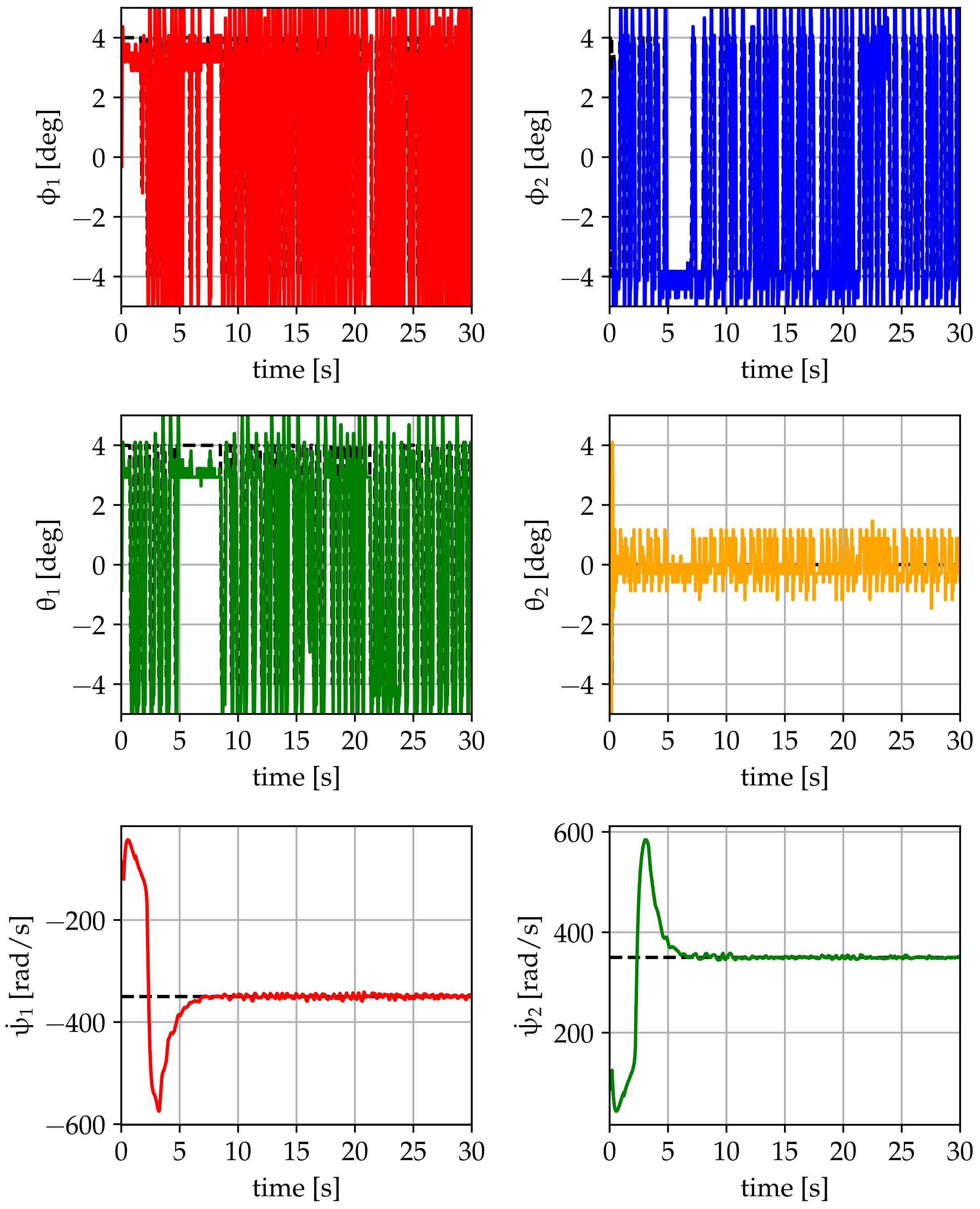


Figure 4.18: Control inputs during trajectory tracking using JPTD algorithm for $\dot{\psi}_1 =$ -350 $\frac{\text{rad}}{\text{s}}$, $\dot{\psi}_2 = 350\ \frac{\text{rad}}{\text{s}}$, $\mathbf{K}_1 = 1000 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 1000 \cdot \mathbf{I}$

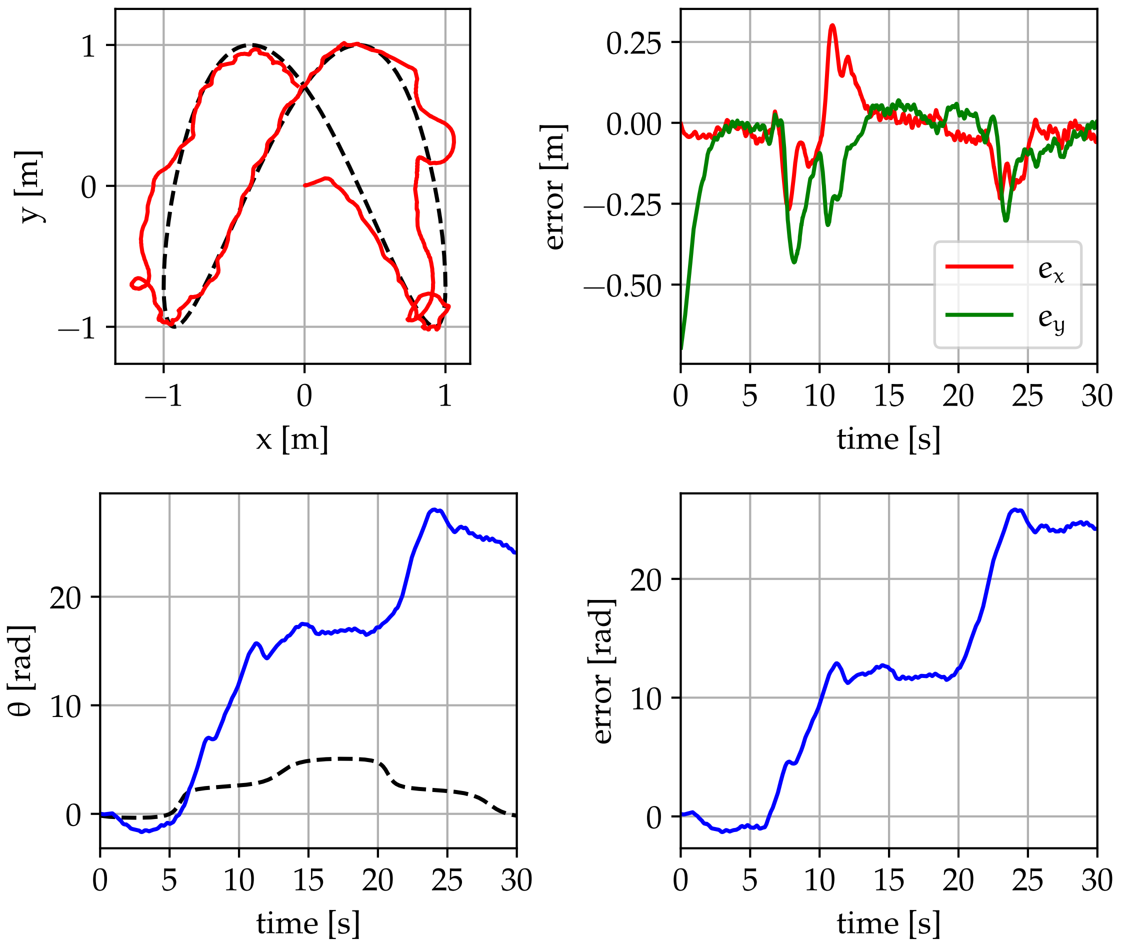


Figure 4.19: Results of trajectory tracking using JPTD algorithm for $\dot{\psi}_1 = \text{-}500\ \frac{\text{rad}}{\text{s}}$, $\dot{\psi}_2 = 500\ \frac{\text{rad}}{\text{s}}$, $\mathbf{K}_1 = 1000 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 1000 \cdot \mathbf{I}$

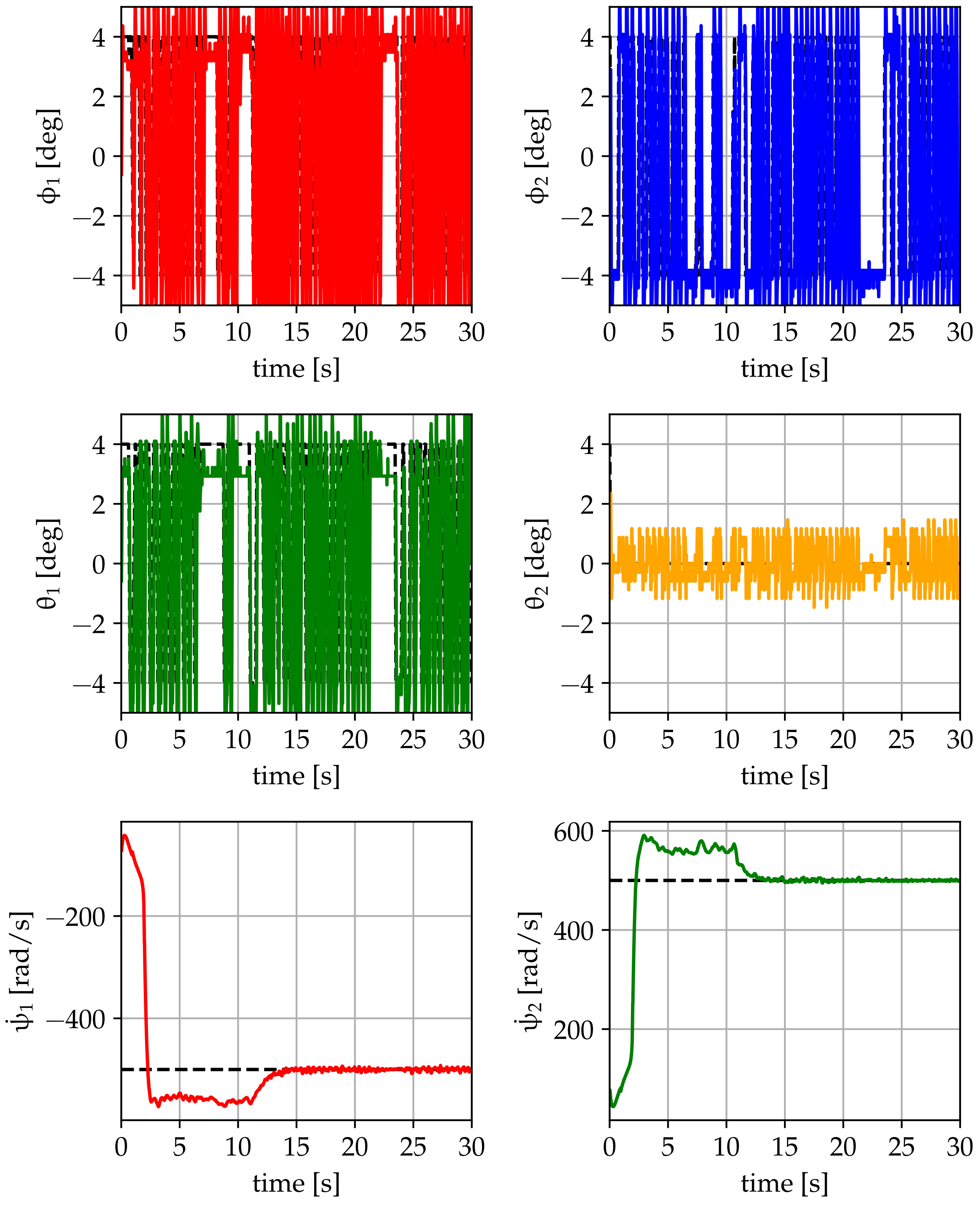


Figure 4.20: Control inputs during trajectory tracking using JPTD algorithm for $\dot{\psi}_1 = $ -500 $\frac{\text{rad}}{\text{s}}$, $\dot{\psi}_2 = 500\ \frac{\text{rad}}{\text{s}}$, $\mathbf{K}_1 = 1000 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 1000 \cdot \mathbf{I}$

Controller	Lower Bound	Upper Bound
inner x	-1.5	1.5
inner y	-1.5	1.5
inner θ	-4π	4π
outer x	-0.5	0.5
outer y	-0.5	0.5
outer θ	$-\pi$	π

Table 4.1: Antiwindup constrains used in cascade PID control

4.3 JPTD–1D Algorithm

This section presents the results of experiments conducted using the JPTD–1D algorithm (see Section 3.3), as shown in Figures 4.21–4.26. The length of the virtual shaft was set to $d = 0.1$ m. The control signals were clamped to the range of $\pm 4^\circ$.

Due to the tracking task being defined for the virtual shaft endpoint, the robot's position follows the trajectory accordingly, while its orientation remains uncontrolled, occasionally performing full $\pm 360^\circ$ rotations. The smallest values of the gain matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ that ensured convergence were found to be 100 and 50, respectively. Again, the integrators saturated immediately, causing high-frequency switching of the angle setpoints between their range limits.

4.4 Cascade PID Control

Finally, the cascade PID control strategy was applied (described in Section 3.4). The results of its tests are shown in in Figures 4.27–4.32. The tuning parameters for the inner and outer loop controllers are provided in Appendix C. The antiwindup constrains was set accordingly to the Table 4.1. The control signals were constrained to the range of $\pm 10^\circ$, corresponding to the maximum allowable operating range defined by the mechanical limitations of the gimbal assembly.

The trajectory tracking performance was the best among all considered algorithms in terms of both the robot's position and orientation, most likely due to the inclusion of basic system dynamics in the control calculations. However, during motion, the robot experienced significant oscillations along the Y_B axis. Nevertheless, due to the proposed transformations, the tracking performance remains unaffected by the angular velocity of the hemispheres. The ϕ_2 gimbal angle undergoes the most significant variations, as it is responsible for controlling both linear motion along the X_B axis and rotational motion around the Z_B axis. The amplitude of the gimbal angles decreases as the hemisphere rotational speed increases.

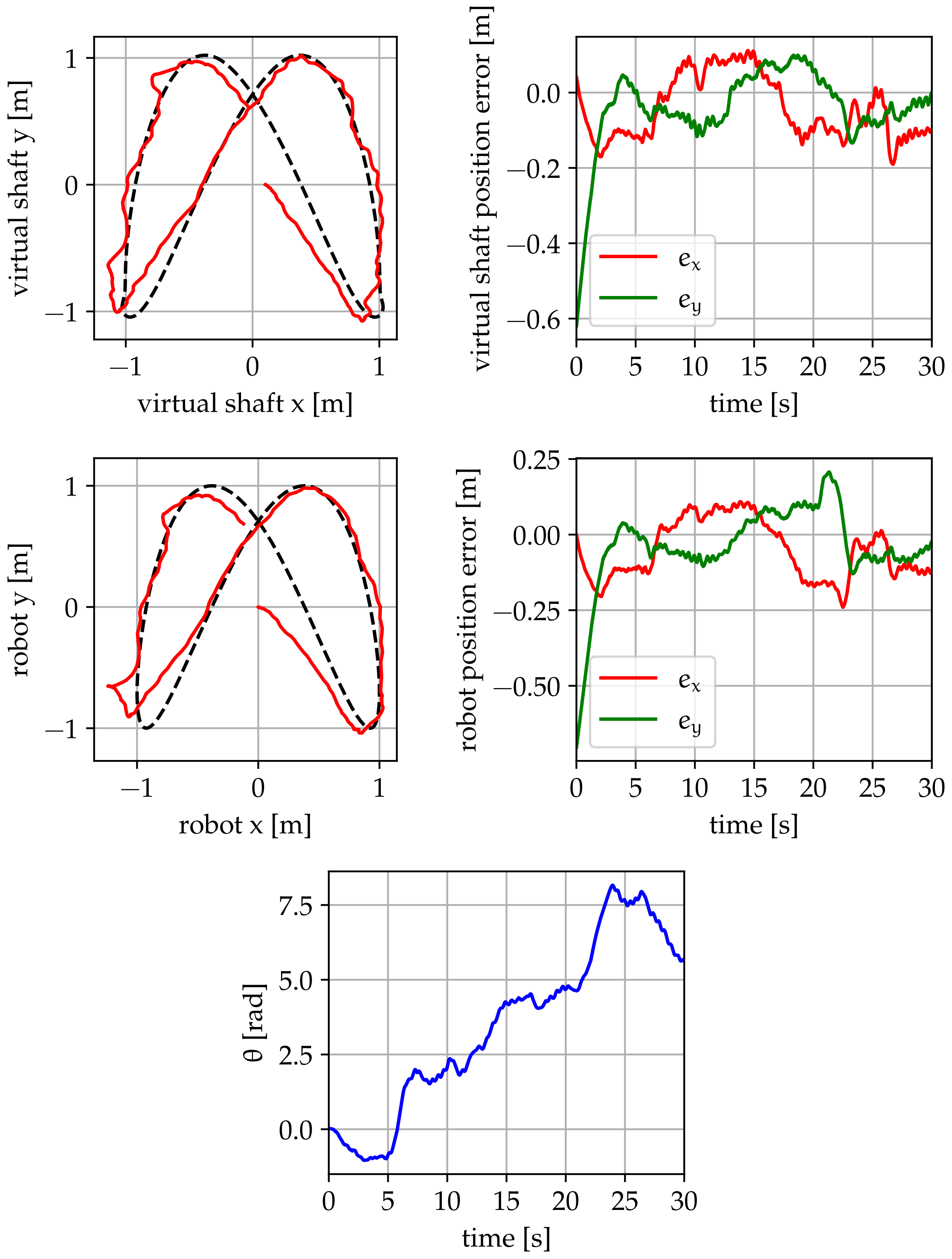


Figure 4.21: Results of trajectory tracking using JPTD–1D algorithm for $\dot{\psi} = -200 \, \frac{\text{rad}}{\text{s}}$, $d = 0.1$ m, $\mathbf{K}_1 = 100 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 50 \cdot \mathbf{I}$

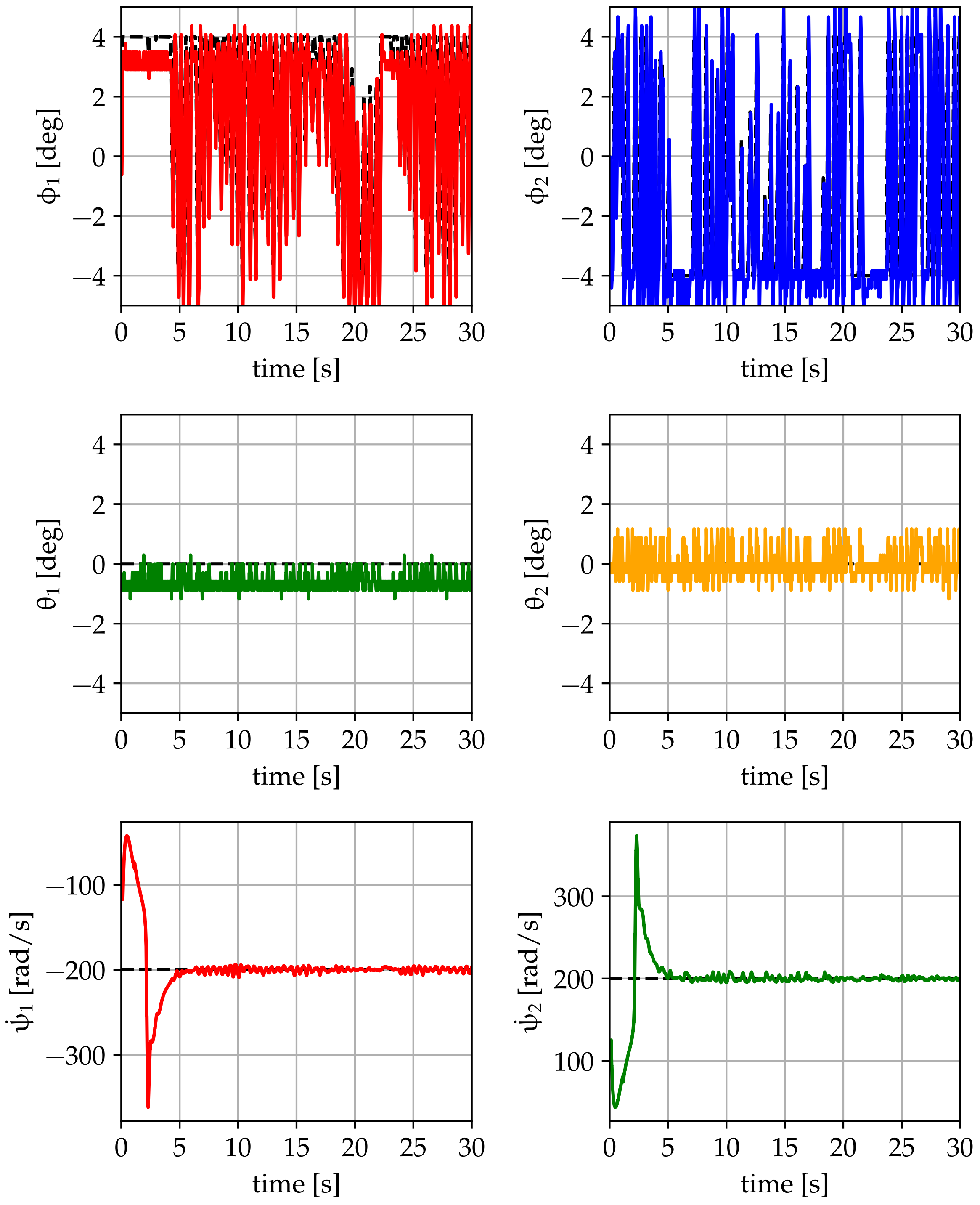


Figure 4.22: Control inputs during trajectory tracking using JPTD–1D algorithm for $\dot{\psi}$ = -200 $\frac{\text{rad}}{\text{s}}$, d = 0.1 m, $\mathbf{K}_1 = 100 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 50 \cdot \mathbf{I}$

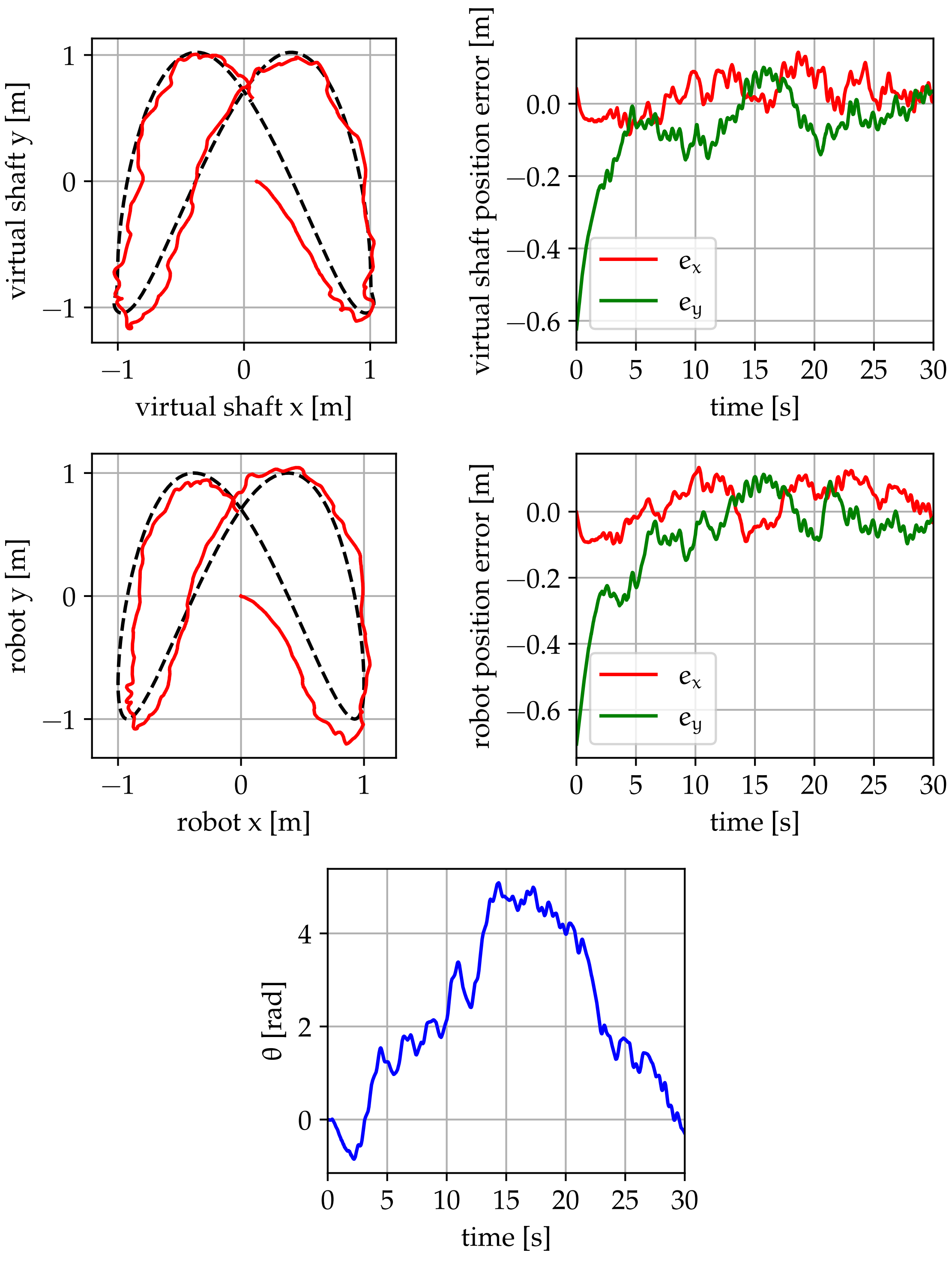


Figure 4.23: Results of trajectory tracking using JPTD–1D algorithm for $\dot{\psi} = -350\ \frac{\text{rad}}{\text{s}}$, d = 0.1 m, $\mathbf{K}_1 = 100 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 50 \cdot \mathbf{I}$

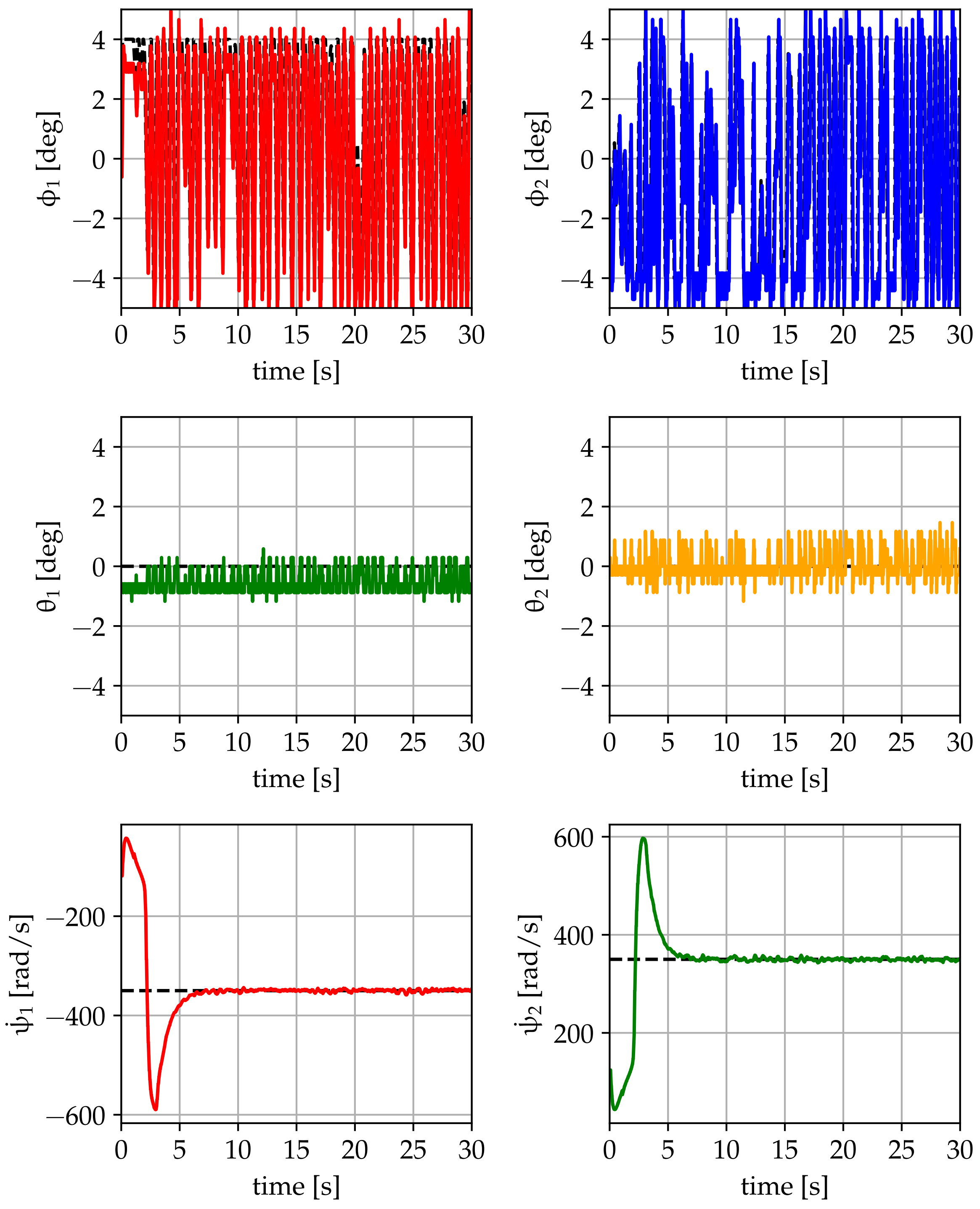


Figure 4.24: Control inputs during trajectory tracking using JPTD–1D algorithm for $\dot{\psi}$ = -350 $\frac{\text{rad}}{\text{s}}$, d = 0.1 m, $\mathbf{K}_1 = 100 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 50 \cdot \mathbf{I}$

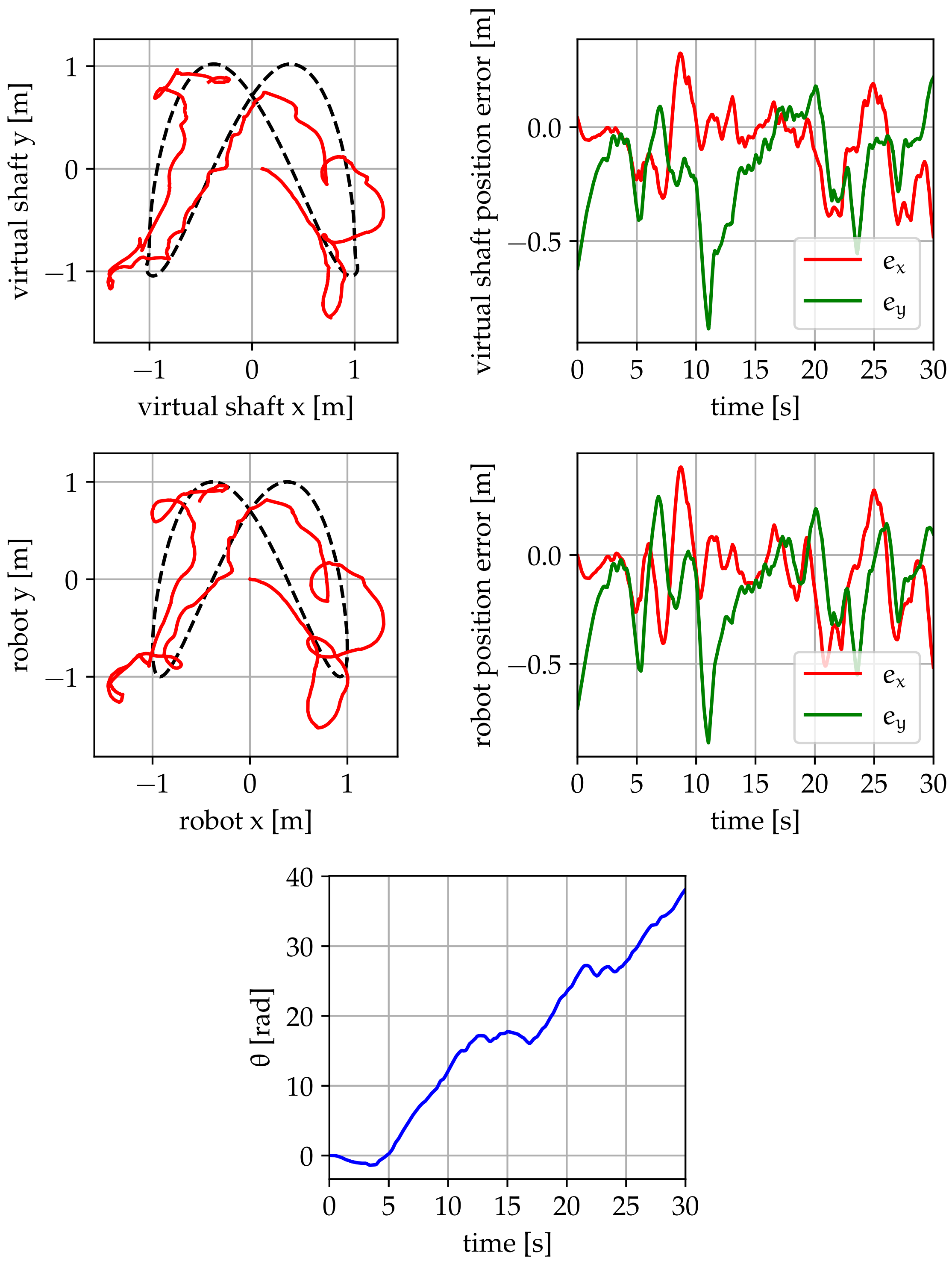


Figure 4.25: Results of trajectory tracking using JPTD–1D algorithm for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$, d = 0.1 m, $\mathbf{K}_1 = 100 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 50 \cdot \mathbf{I}$

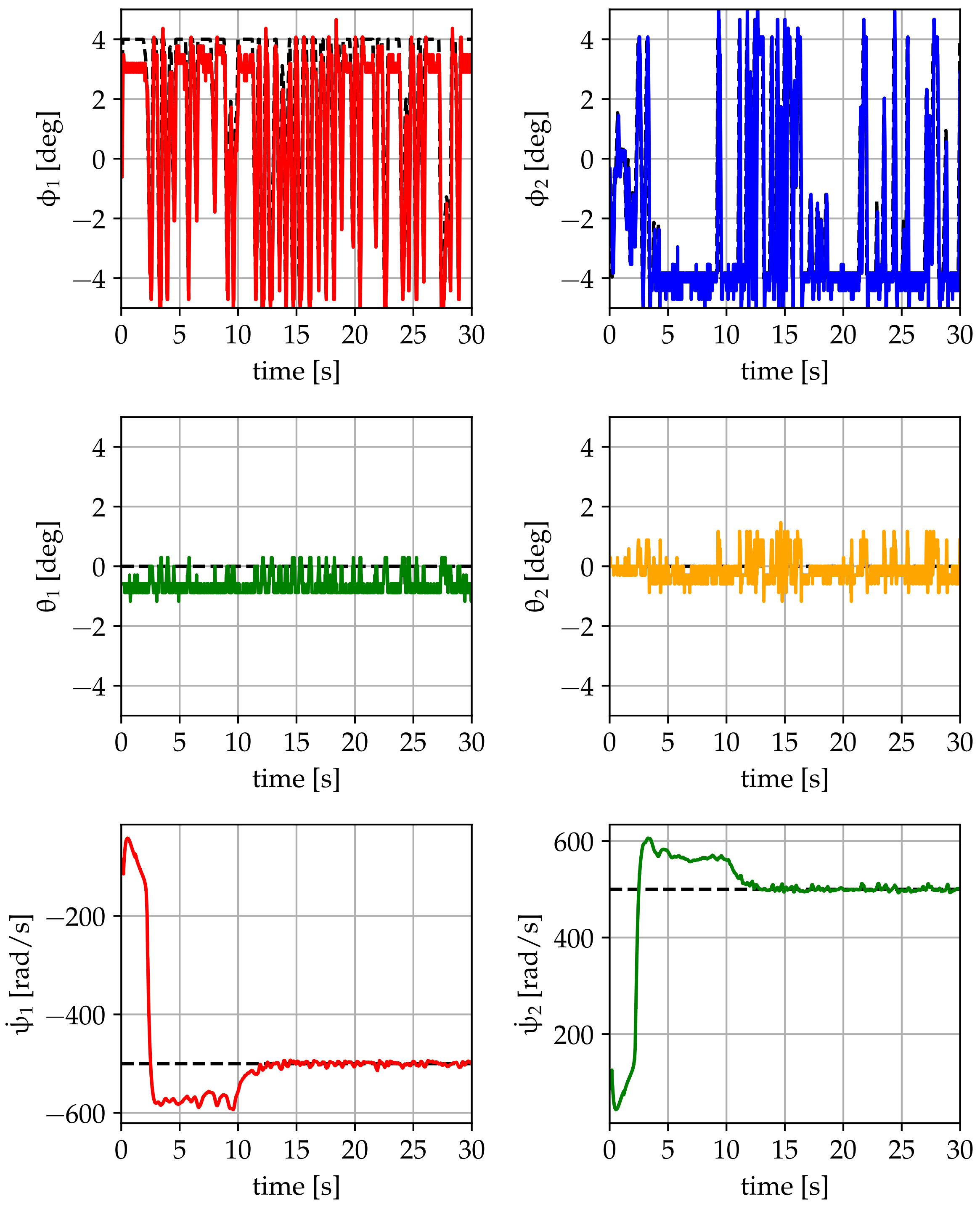


Figure 4.26: Control inputs during trajectory tracking using JPTD–1D algorithm for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$, d = 0.1 m, $\mathbf{K}_1 = 100 \cdot \mathbf{I}$, and $\mathbf{K}_2 = 50 \cdot \mathbf{I}$

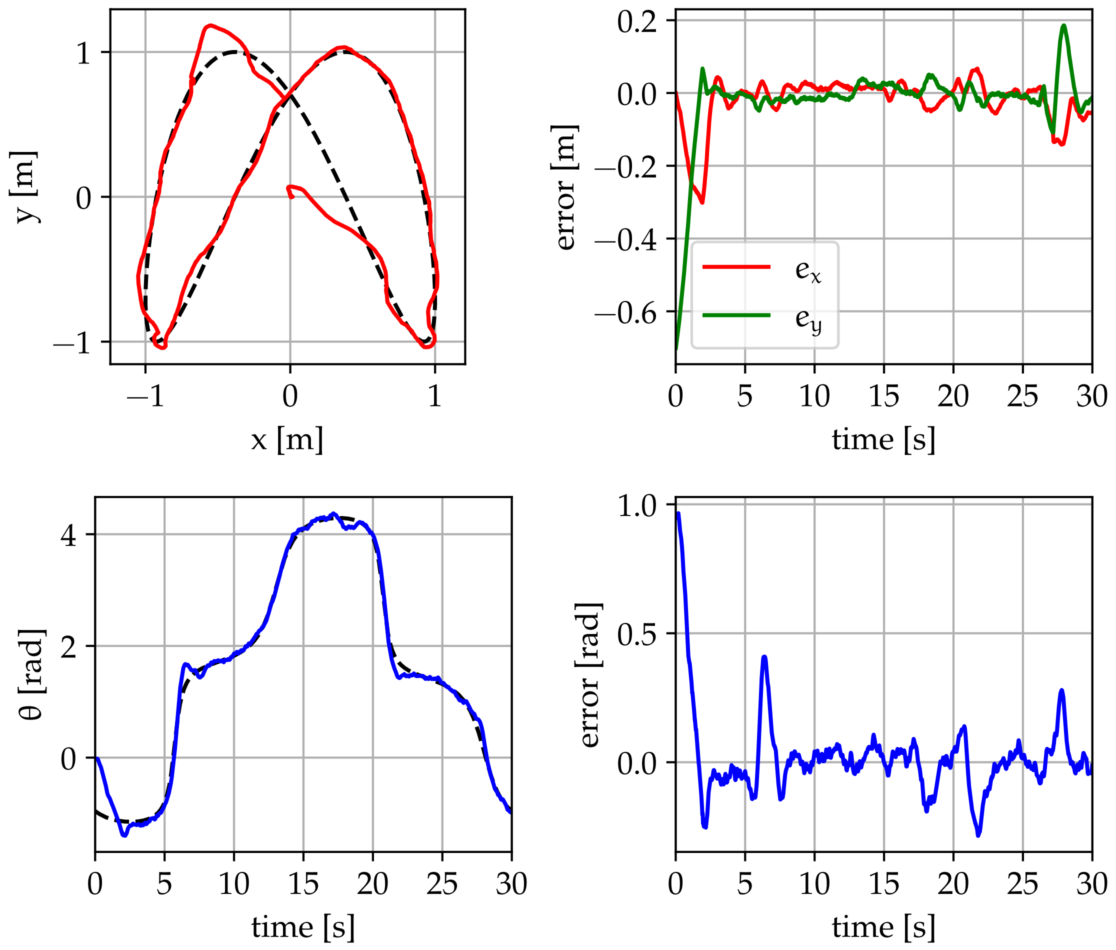


Figure 4.27: Results of trajectory tracking using cascade PID control for $\dot{\psi}$ = -200 $\frac{\text{rad}}{\text{s}}$

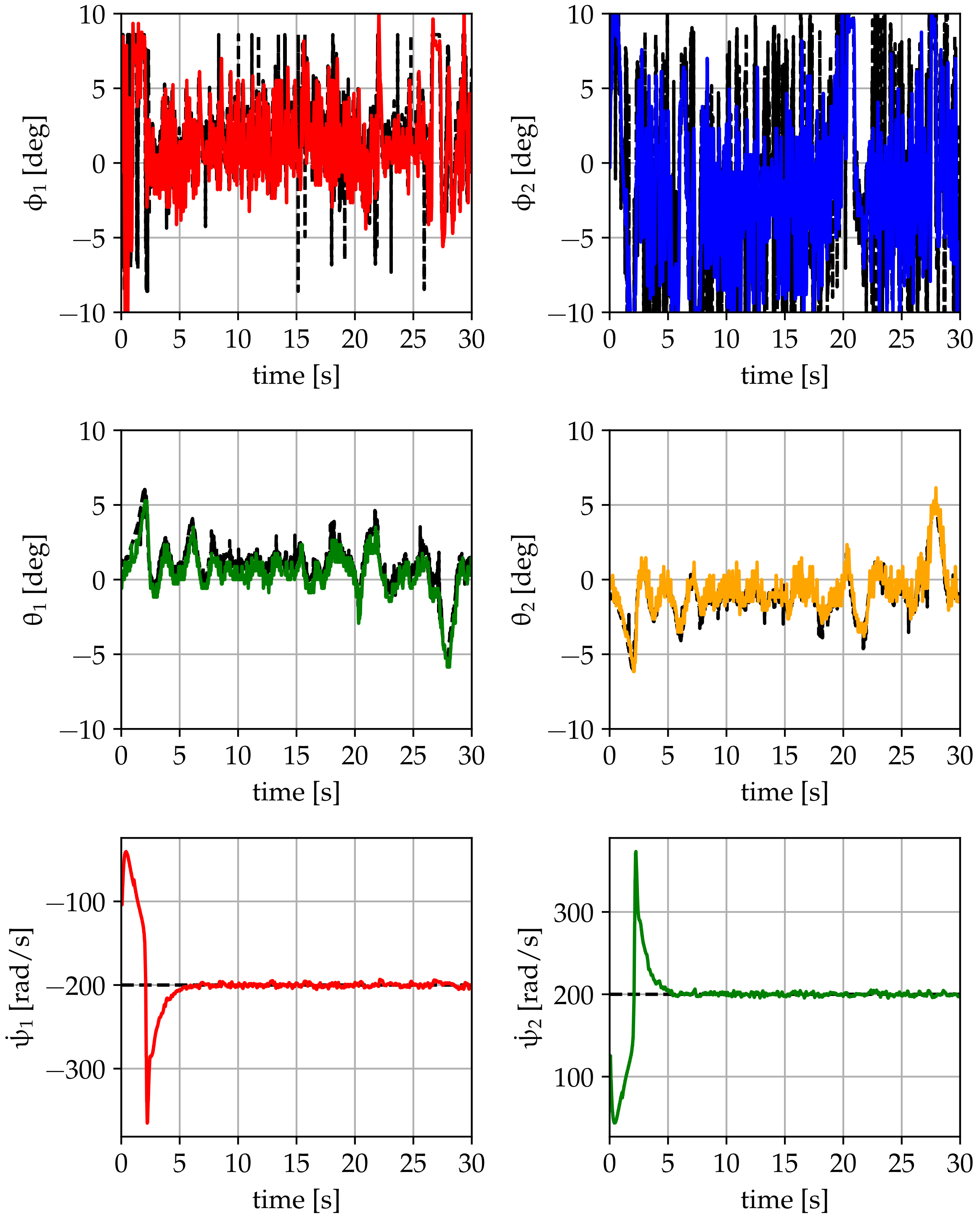


Figure 4.28: Control inputs during trajectory tracking using cascade PID control for $\dot{\psi}$ = -200 $\frac{\text{rad}}{\text{s}}$

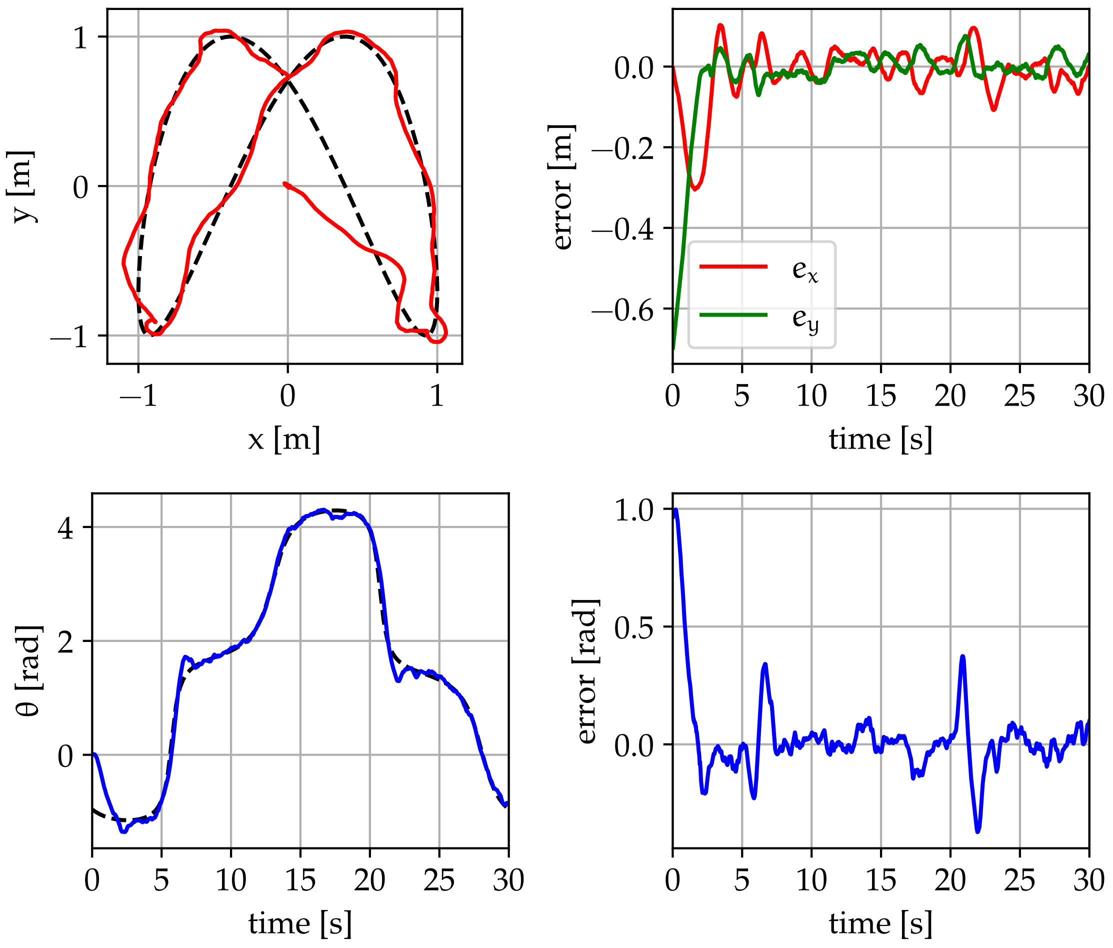


Figure 4.29: Results of trajectory tracking using cascade PID control for $\dot{\psi}$ = -350 $\frac{\text{rad}}{\text{s}}$

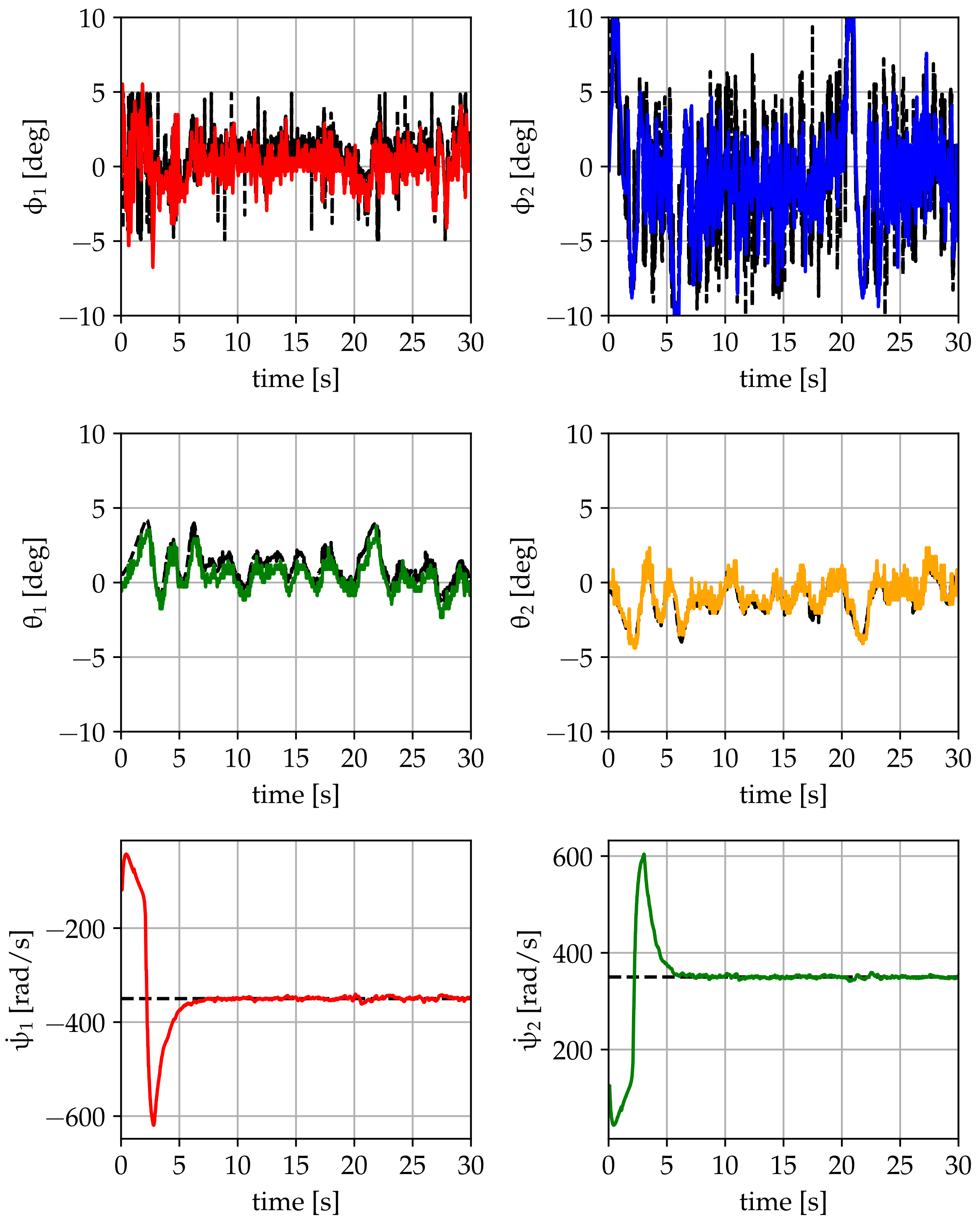


Figure 4.30: Control inputs during trajectory tracking using cascade PID control for $\dot{\psi}$ = -350 $\frac{\text{rad}}{\text{s}}$

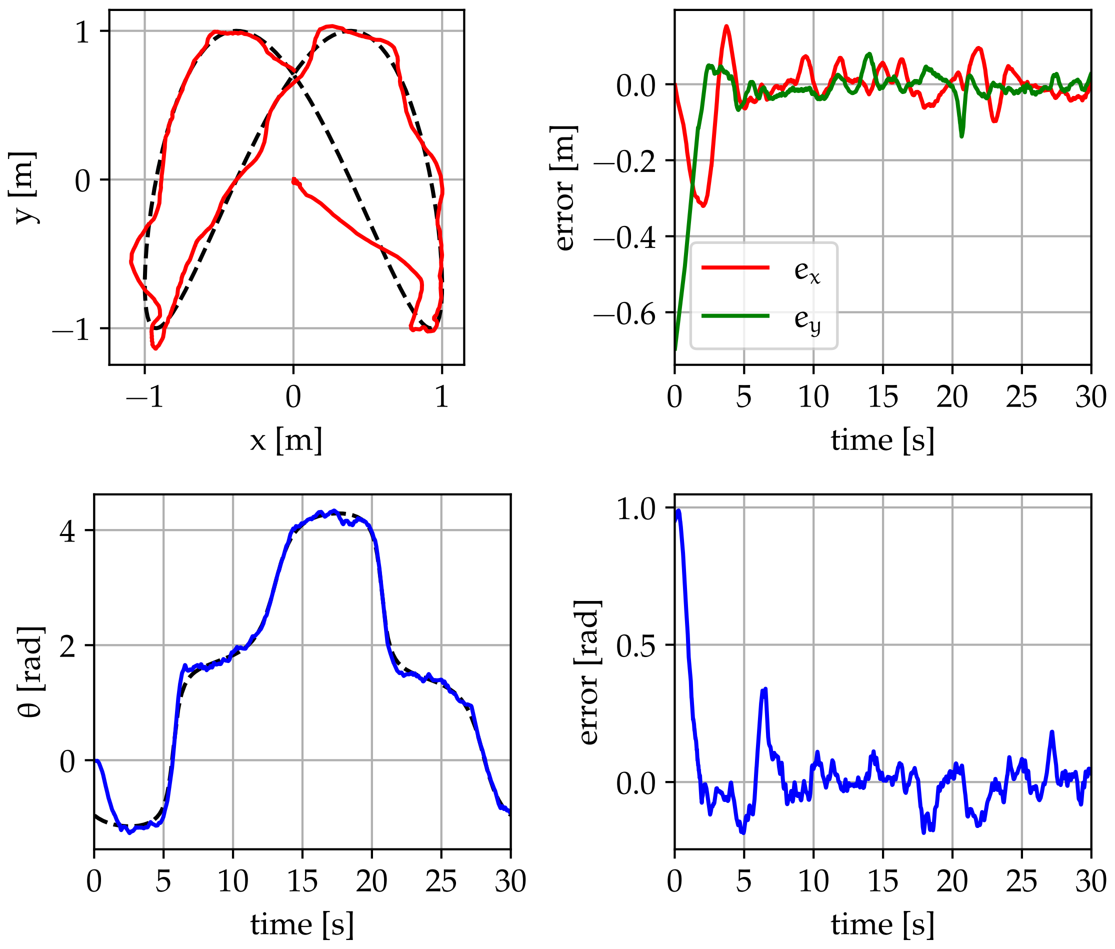


Figure 4.31: Results of trajectory tracking using cascade PID control for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$

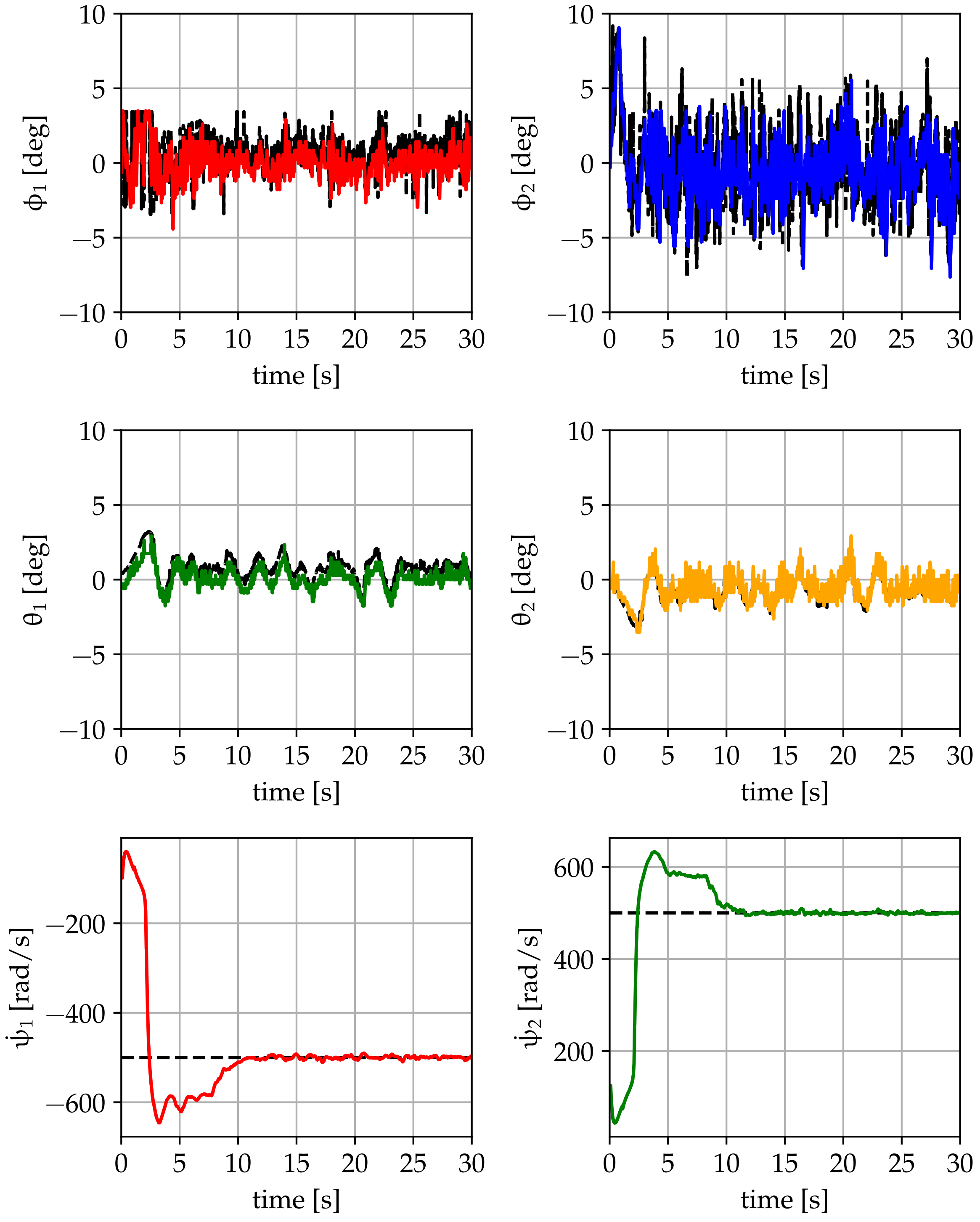


Figure 4.32: Control inputs during trajectory tracking using cascade PID control for $\dot{\psi}$ = -500 $\frac{\text{rad}}{\text{s}}$

Algorithm		**Hemispheres Angular Velocity**		
		200 $\frac{rad}{s}$	**350 $\frac{rad}{s}$**	**500 $\frac{rad}{s}$**
Simultaneous (identical)	XY	big oscillations	big oscillations	big oscillations
	θ	fast spin	fast spin	fast spin
Simultaneous (mirrored)	XY	oscillations	oscillations	oscillations
	θ	slow spin	slow spin	slow spin
JPTD	XY	good	good	good
	θ	constant error	not converge	not converge
JPTD–1D	XY	good	good	big oscillations
	θ	oscillations	oscillations	fast spin
Cascade PID	XY	good	good	good
	θ	good	good	good

Table 4.2: Overview of algorithms performance

4.5 Results Overview

The results of the experiments conducted in this chapter are summarized in Table 4.2. The table presents the overall performance of the algorithms in tracking the reference position (XY) and orientation (θ) for the given hemispheres angular velocities. Among all evaluated algorithms, JPTD, JPTD–1D, and cascade PID strategy demonstrated good performance in tracking the reference position, however only the PID cascade was able to reliably control the robot's orientation.

Unfortunately, due to environmental constraints, the true position and orientation of the robot were not recorded. Nevertheless, during numerous test runs, it was frequently observed that the estimated state deviated from the robot's actual state, particularly during high-speed maneuvers. With increase of hemispheres angular velocity, the overall performance of control strategies relying only on the robot's kinematics declined.

Chapter 5

Summary

The goal of this thesis was to modernize the existing mechanical design a mobile robot equipped with two HOG wheels, develop a control system for such a robot, and to utilize it for the analysis of selected trajectory tracking control algorithms. A dedicated hardware platform, including a custom controller module, was developed alongside supporting software for communication, data acquisition, sensor fusion, and the implementation of multiple control strategies. The objectives outlined in the work have been successfully achieved.

The mechanical modifications, along with the integration of new servomotors, significantly enhanced the capabilities of the platform. However, the design of the gimbal mechanism still exhibits several limitations, such as the relatively high mass of the links compared to 2 DOF spherical parallel manipulators. The overall mechanical backlash remain highly dependent on the internal servo gear quality. Vibrations occurring during operation contribute to the accelerated wear of these gears, resulting in a noticeable decrease of control precision, even by several degrees per axis. Reducing the radius of the hemispheres would improve the effective resolution of the servomotors and reduce the impact of servo backlash. Additionally, the initial assumption regarding the high rotational speed of the hemispheres proved to be incorrect. The use of a sensorless BEMF method for driving BLDC motors was found to be unsuitable for this application. Furthermore, the operational environment and the maximum velocity of the robot were significantly constrained by the use of an optical flow sensor, which is primarily intended for far-field applications. Relying solely on velocity integration for position estimation, without any external correction mechanisms, resulted in considerable position drift, particularly evident during high-speed motion. In the future, alternative methods for robot localization need to be introduced.

The analysis of the experiments indicates that control strategies based solely on the robot's kinematics exhibit poor performance in practical applications. The cascade PID control, being the only algorithm that incorporates the basic robot's dynamics, achieved the best results. As the angular velocities of the hemispheres increased, the tracking performance of the kinematic controllers decreased. The two tested variants of the simultaneous control strategy demonstrated that the friction between the hemispheres and even a flat ground surface is not negligible. To minimize the influence of this friction, the hemispheres should rotate in opposite directions at equal speeds. One of the

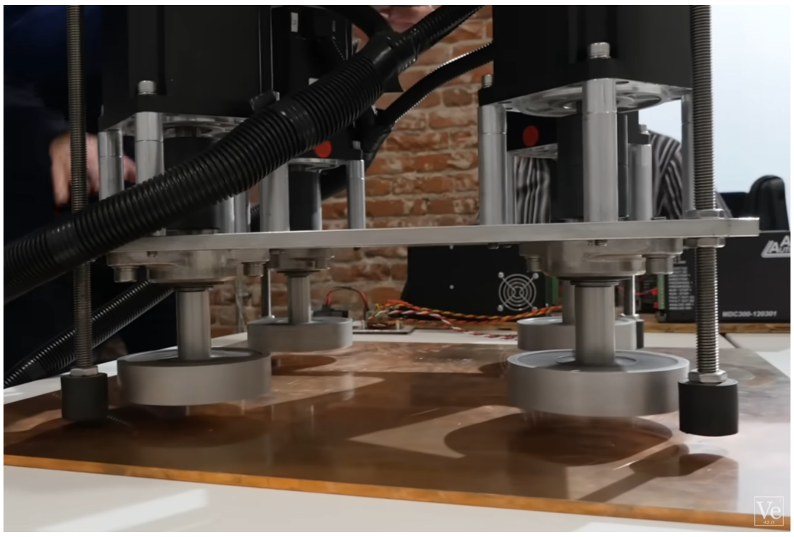


Figure 5.1: Hyperloop One magnetic levitation test platform (2017) [Mul]

conclusions from [Jon17] remain valid: in the future, the dynamic model of the *Hogger*^{2} robot needs to be derived and utilized for the design of control system. It is necessary to develop a physics simulation for testing upcoming control algorithms that includes system dynamics, hemisphere and ground irregularities, as well as friction, using modern frameworks like *MuJoCo* [TET12], *NVIDIA Isaac* [NVI], or even *Drake* [TtDDT19].

The most significant drawback of the HOG drive is the unrealistic assumption of a single point of contact between the hemisphere and the ground. Even small shape imperfections cause noticeable movement errors. During operation, the hemispheres wear unevenly near the rotation axis, which is critical when combined with above.

If the center of gravity were placed below the geometric center of the hemispheres, it would be possible to eliminate the additional support wheel, creating a self-stabilizing system with only two points of contact. Taking this idea further, it might even be feasible to transform the *Hogger*^{2} robot into an amphibious vehicle, provided that the hemispheres are sufficiently buoyant and the center of buoyant force is located above the center of gravity.

It is worth mentioning that there exist magnetic levitation methods [CBK19], which use rotating discs populated with high-strength magnets above copper rails, as shown in Figure 5.1. By applying rotational motion, the magnets generate a magnetic field that acts as an electromagnetic suspension, eliminating friction caused by surface contact. In the future, replacing the hemisphere wheels with a form of magnetic levitation may eliminate issues related to friction and surface irregularities of the hemispheres. An asymmetrical magnetic field generated by magnets rotating around a gimballed axis may be the next evolution stage of the HOG wheel.

Bibliography

[Ack] E. Ackerman. You've Never Seen a Robot Drive System Like This Before. https://spectrum.ieee.org/youve-never-seen-a-drive-system-like-this-before.

[ÅH95] K. J. Åström and T. Hägglund. *PID Controllers*. Setting the standard for automation. International Society for Measurement and Control, 1995.

[Boc19] J. Boczar. Control system for two HOG wheel mobile robot. Master's thesis, https://kcir.pwr.edu.pl/~mucha/Pracki/Jedrzej_Boczar_praca_magisterska.pdf, Wrocław University of Science and Technology, 2019.

[CBK19] E. Chaidez, Shankar P. Bhattacharyya, and Adonios N. Karpetis. Levitation Methods for Use in the Hyperloop High-Speed Transportation System. *Energies*, 12(21), 2019.

[Cha] J. P. Charras. KiCad. https://www.kicad.org/.

[DER] DERUTU Inc. DERUTU Basement polishing robot. https://www.youtube.com/watch?v=O5tWiZWLNQU.

[Fur] S. Furuhashi. MessagePack. https://msgpack.org/index.html.

[Gór17] D. Góral. Konstrukcja robota mobilnego napędzanego dwiema półsferami. Bachelor's thesis, https://kcir.pwr.edu.pl/~mucha/Pracki/Damian_Goral_praca_inzynierska.pdf, Wrocław University of Science and Technology, 2017. (in Polish).

[Gro] Qt Group. Qt. https://www.qt.io/.

[Gun] C. Gunyon. An implementation of the MessagePack serialization format in C. https://github.com/camgunz/cmp.

[Hon] Honeywell. Three-Axis Digital Compass IC HMC5883L. https://www.farnell.com/datasheets/1683374.pdf.

[ifd] ifdesign.com. Basement Concrete Troweling and Leveling Robot. https://ifdesign.com/en/winner-ranking/project/basement-concrete-troweling-and-leveling-robot/314447.

[Inf] Infineon. OptiMOS-T2 Power-Transistor. https://www.infineon.com/dgdl/Infineon-IPD90N04S4L_04-DS-v01_00-en.pdf?fileId=db3a304328c6bd5c01291cab92685eaa.

[Ins] Texas Instruments. LM2105 107-V, 0.5-A, 0.8-A Half-Bridge Driver with 5-V UVLO and Integrated Bootstrap Diode. https://www.ti.com/lit/ds/symlink/lm2105.pdf?ts=1683842340255&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM2105.

[Inv] InvenSense. MPU-6000 and MPU-6050 Product Specification Revision 3.4. https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[Jac21] E. B. Jacobsen. Modelling and Control of Thrust Vectoring Mono-copter. Master's thesis, https://projekter.aau.dk/projekter/files/421577367/Master_Thesis_Emil_Jacobsen_v5.pdf, Aalborg University, 2021.

[JG] B. Jacob and G. Guennebaud. Eigen. https://eigen.tuxfamily.org/index.php?title=Main_Page.

[Jon14] P. Joniak. Badania symulacyjne zachowania robota mobilnego napędzanego dwiema półsferami. Bachelor's thesis, https://kcir.pwr.edu.pl/~mucha/Pracki/Pawel_Joniak_projekt_inzynierski.pdf, Wrocław University of Science and Technology, 2014. (in Polish).

[Jon17] P. Joniak. Zadanie sterowania robota mobilnego napędzanego dwiema półsferami. Master's thesis, https://kcir.pwr.edu.pl/~mucha/Pracki/Pawel_Joniak_praca_magisterska.pdf, Wrocław University of Science and Technology, 2017. (in Polish).

[KAV] KAVAN. C2830-750. https://kavanrc.com/en/item/kavan-brushless-motor-c2830-750-140796.

[KCB+23] M. Kabiri, C. Cimarelli, H. Bavle, J. L. Sanchez-Lopez, and H. Voos. A Review of Radio Frequency Based Localisation for Aerial and Ground Robots with 5G Future Perspectives. *Sensors*, 23(1), 2023.

[KSBT+17] J. Krishnaraj, K. Sangeetha, M. V. Babu Tanneru, V. V. S. Prasad Harnadh, and M. Vishnu Vardhan. A Mecanum Wheel Based Robot Platform for Warehouse Automation. *International Journal of Mechanical Engineering and Technology*, 8:181–189, 07 2017.

[LCA] A. Lita, M. Cheles, and A. Abacan. Sensorless BLDC Control with Back-EMF Filtering Using a Majority Function. https://ww1.microchip.com/downloads/aemDocuments/documents/MCU16/ApplicationNotes/ApplicationNotes/AN1160-Sensorless-BLDC-Control-with-Back-EMF-Filtering-Using-a-Majority-Function-DS00001160.pdf.

[Lub23] T. Lubelski. Budowa małego mobilnego robota laboratoryjnego klasy (1,2). Bachelor's thesis, https://kcir.pwr.edu.pl/~mucha/Pracki/Tomek_Lubelski_praca_inzynierska.pdf, Wrocław University of Science and Technology, 2023. (in Polish).

[mec38] Hemisphere Drive Speedster. *Mechanics and Handicraft*, 5(9), 1938. https://web.archive.org/web/20201201092333/http://blog.modernmechanix.com/hemisphere-drive-speedster/#more.

[Moż24] E. Możdżeń. Controller for Hogger^2 HOG wheel robot – project proposal. Report, https://github.com/Eryk-Mozdzen/hogger2/blob/master/docs/intermediate_project_report_1.pdf, Wrocław University of Science and Technology, 2024.

[Moż25] E. Możdżeń. Controller for Hogger^2 HOG wheel robot – final report. Report, https://github.com/Eryk-Mozdzen/hogger2/blob/master/docs/intermediate_project_report_final.pdf, Wrocław University of Science and Technology, 2025.

[Mul] D. Muller. Electromagnetic Levitation Quadcopter. https://www.youtube.com/watch?v=pCON4zfMzjU.

[NVI] NVIDIA Corporation. NVIDIA Isaac. https://developer.nvidia.com/isaac.

[Ozy] T. Ozyagcilar. Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference. https://www.nxp.com/docs/en/application-note/AN4246.pdf.

[Pix] PixArt Imaging Inc. PMW3901MB-TXQT: Optical Motion Tracking Chip. https://wiki.bitcraze.io/_media/projects:crazyflie2:expansionboards:pot0189-pmw3901mb-txqt-ds-r1.00-200317_20170331160807_public.pdf.

[ROBa] ROBOBLOCK. AI Concrete Slab Finisher Robot. https://www.youtube.com/watch?v=kyBuMxqpy-o.

[ROBb] ROBOTICS. Dynamixel AX-12A eManual. https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/.

[Ryb13] M. Rybczyński. Model robota mobilnego napędzanego za pośrednictwem półsfery. Bachelor's thesis, https://kcir.pwr.edu.pl/~mucha/Pracki/Michal_Rybczynski_praca_inzynierska.pdf, Wrocław University of Science and Technology, 2013. (in Polish).

[STM] STMicroelectronics. RM0481 Reference manual. https://www.st.com/resource/en/reference_manual/rm0481-stm32h52333xx-stm32h56263xx-and-stm32h573xx-armbased-32bit-mcus-stmicroelectronics.pdf.

[TET12] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[TM17] K. Tchoń and R. Muszyński. *Mathematical Methods of Automation and Robotics*. 2017. https://kcir.pwr.edu.pl/~mucha/Skrypty/KTRM_MMAiR_eng.pdf.

[TtDDT19] R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics. https://drake.mit.edu, 2019.

[Wika] Wikipedia. Extended Kalman filter. https://en.wikipedia.org/wiki/Extended_Kalman_filter.

[Wikb] Wikipedia. JSON. https://en.wikipedia.org/wiki/JSON.

[Wikc] Wikipedia. Singular value decomposition. https://en.wikipedia.org/wiki/Singular_value_decomposition.

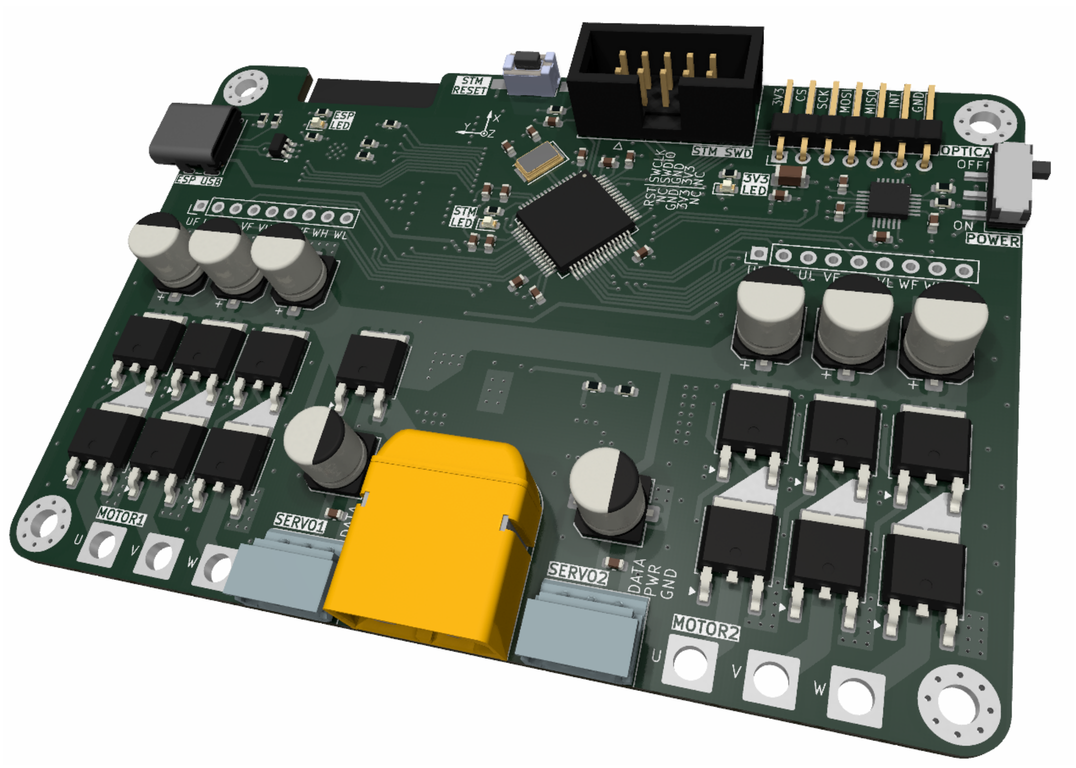
[zer] ZeroMQ. https://zeromq.org/.

Appendix A

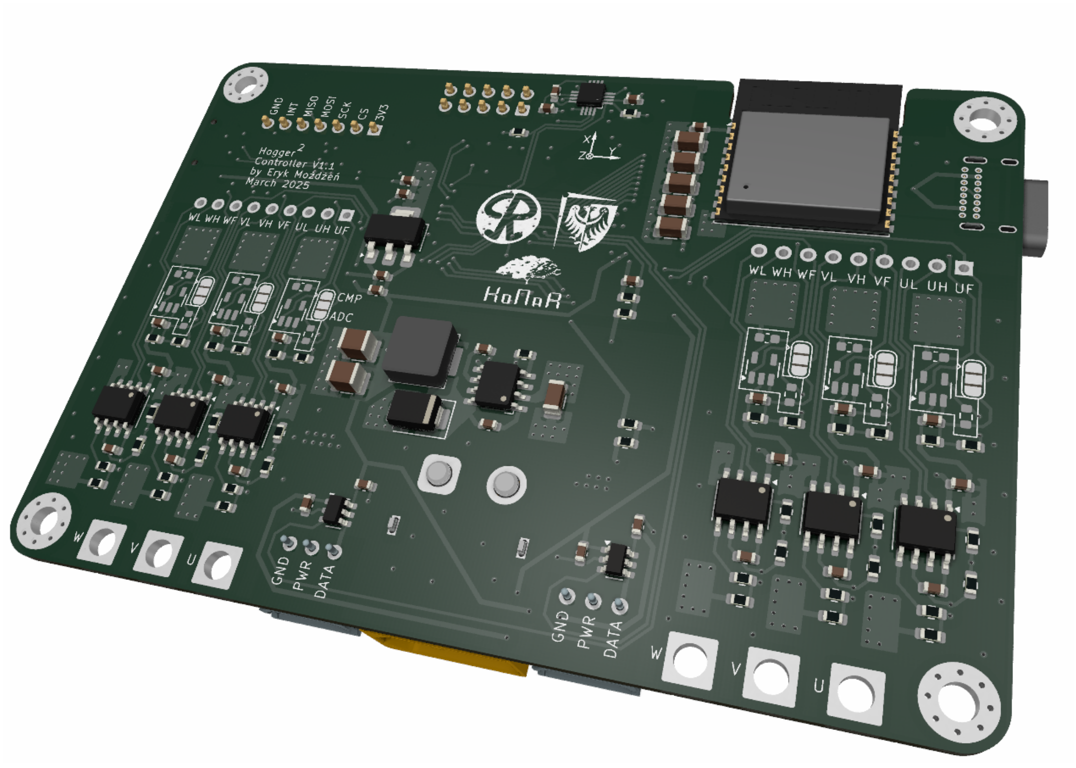
Electronic Schematic

This appendix contains the electronic schematics and PCB images of the controller module[*]. The design was created using the KiCad software [Cha].

[*]The complete hardware design, firmware source code, code generation tools, endpoint applications, and supporting utilities, along with previously unpublished materials provided by the authors of [Jon14, Jon17, Boc19], are available in a public Git repository https://github.com/Eryk-Mozdzen/hogger2.



(a) Top view


GND INT MISO MOSI SCK CS 3V3
Hogger² Controller V1.1 by Eryk Możdżeń March 2025
WL WH WF VL VH VF UL UH UF
CMP
ADC
KoNaR
GND PWR DATA

(b) Bottom view

Figure A.1: PCB design

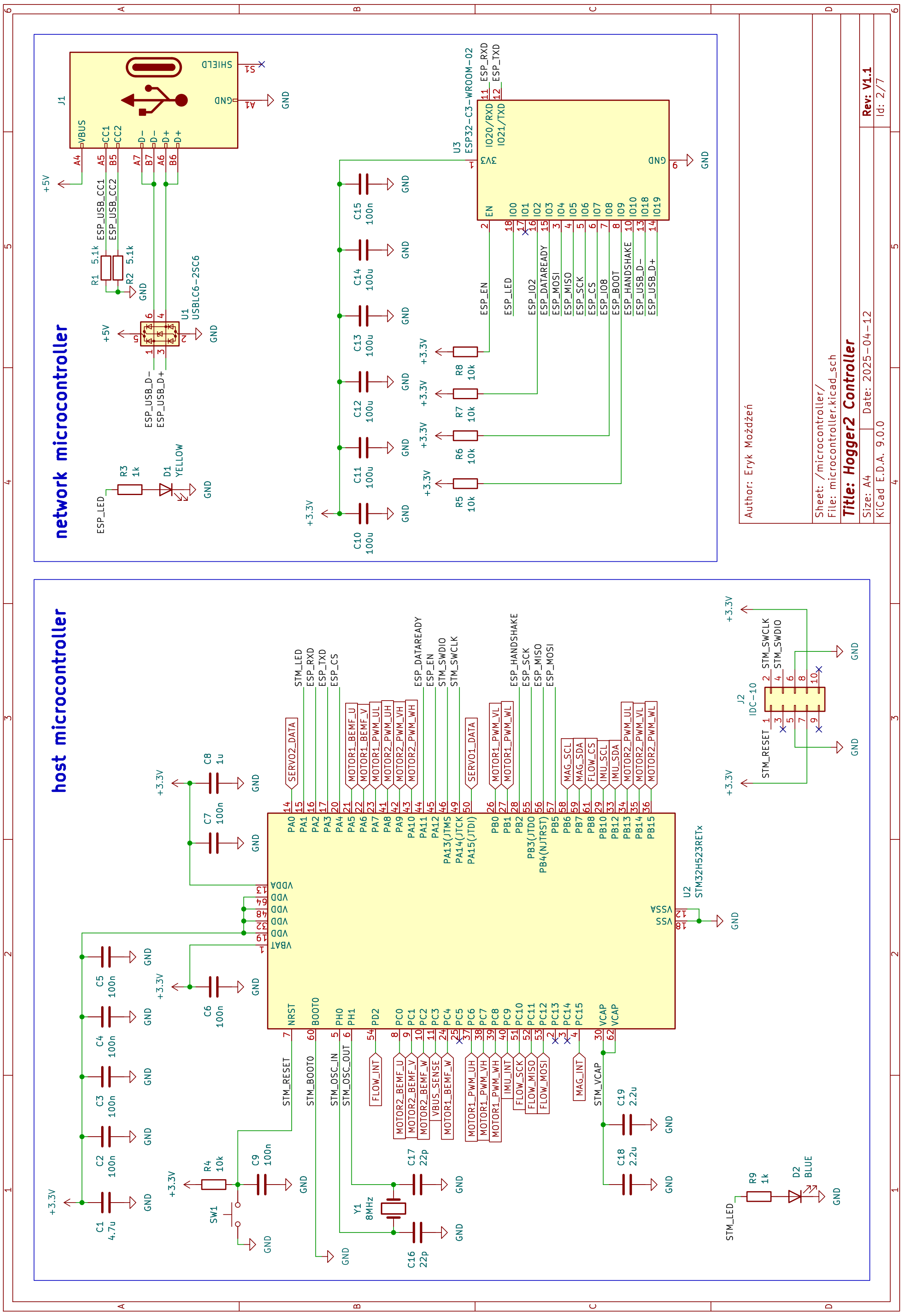


Figure A.2: Host and communication microcontrollers

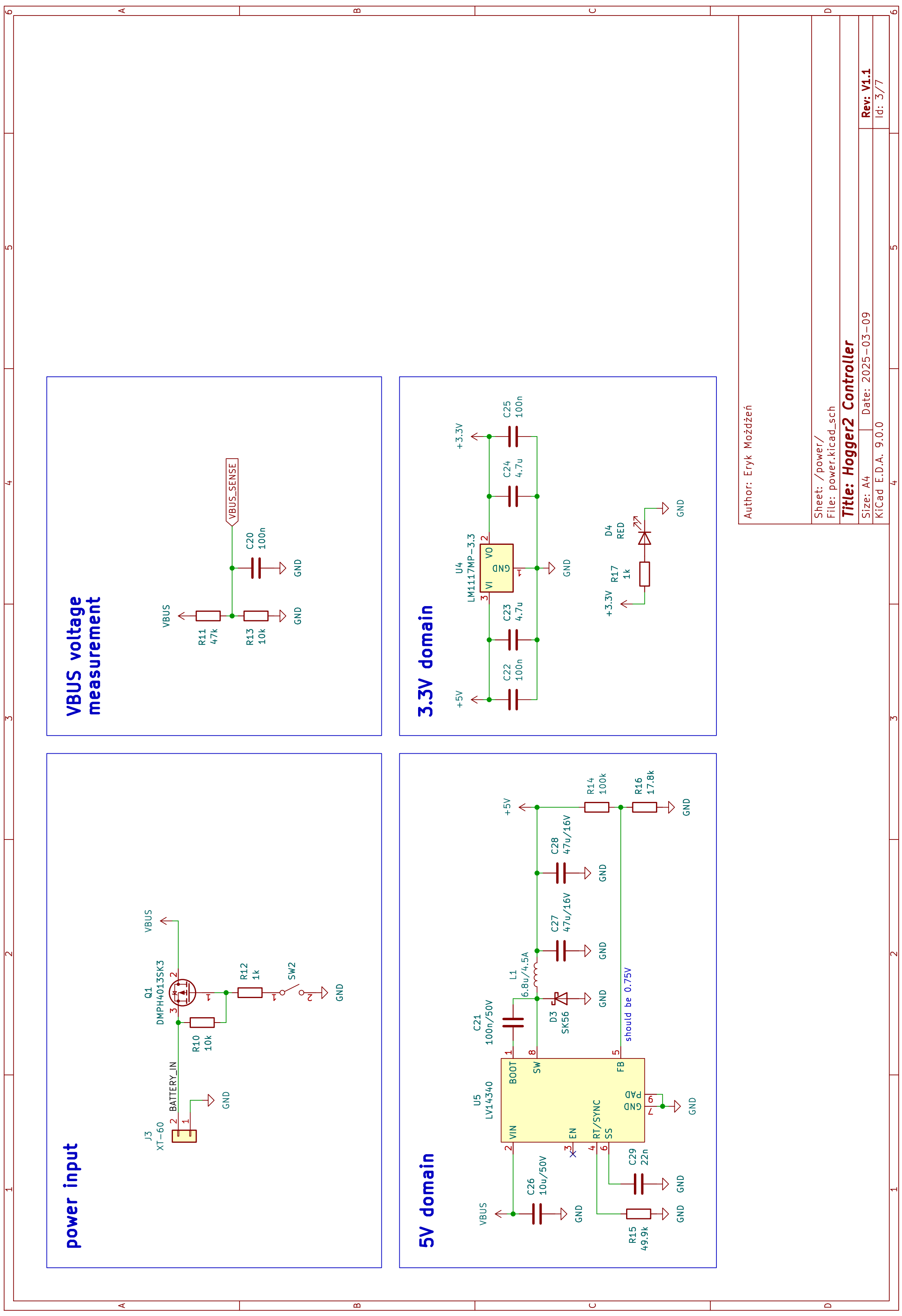


Figure A.3: Power circuits

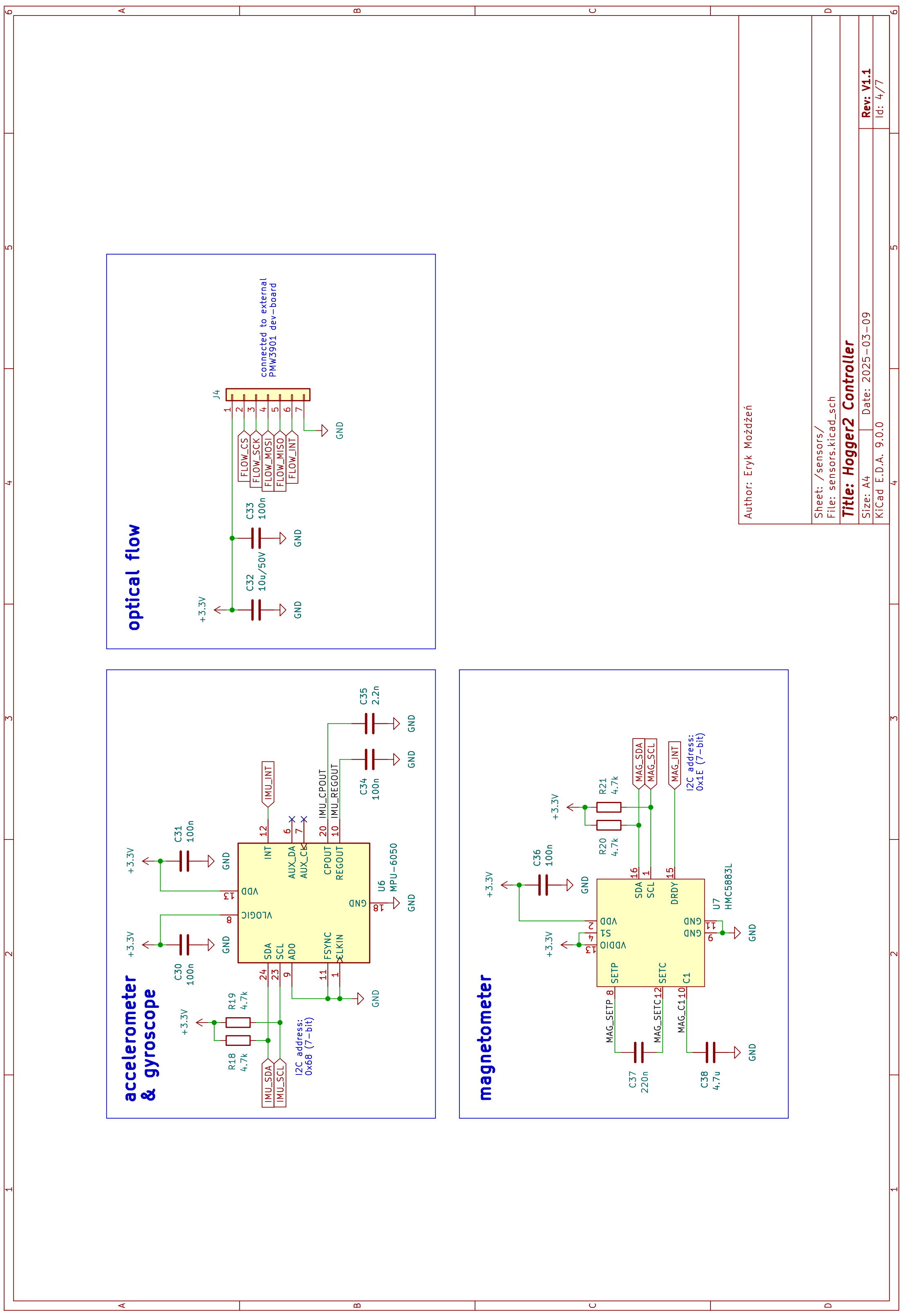


Figure A.4: Sensor circuits

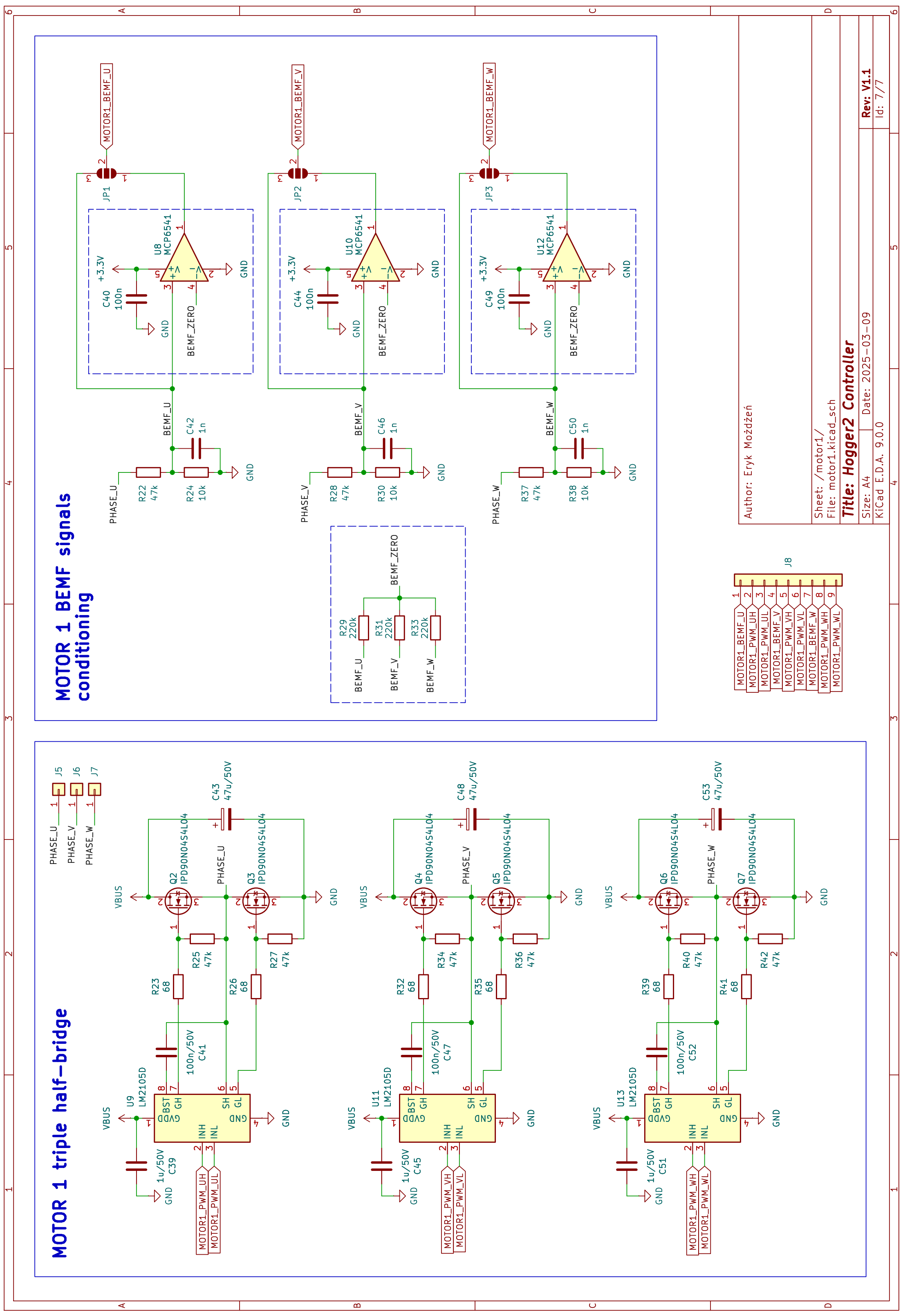


Figure A.5: Motor 1 control circuits

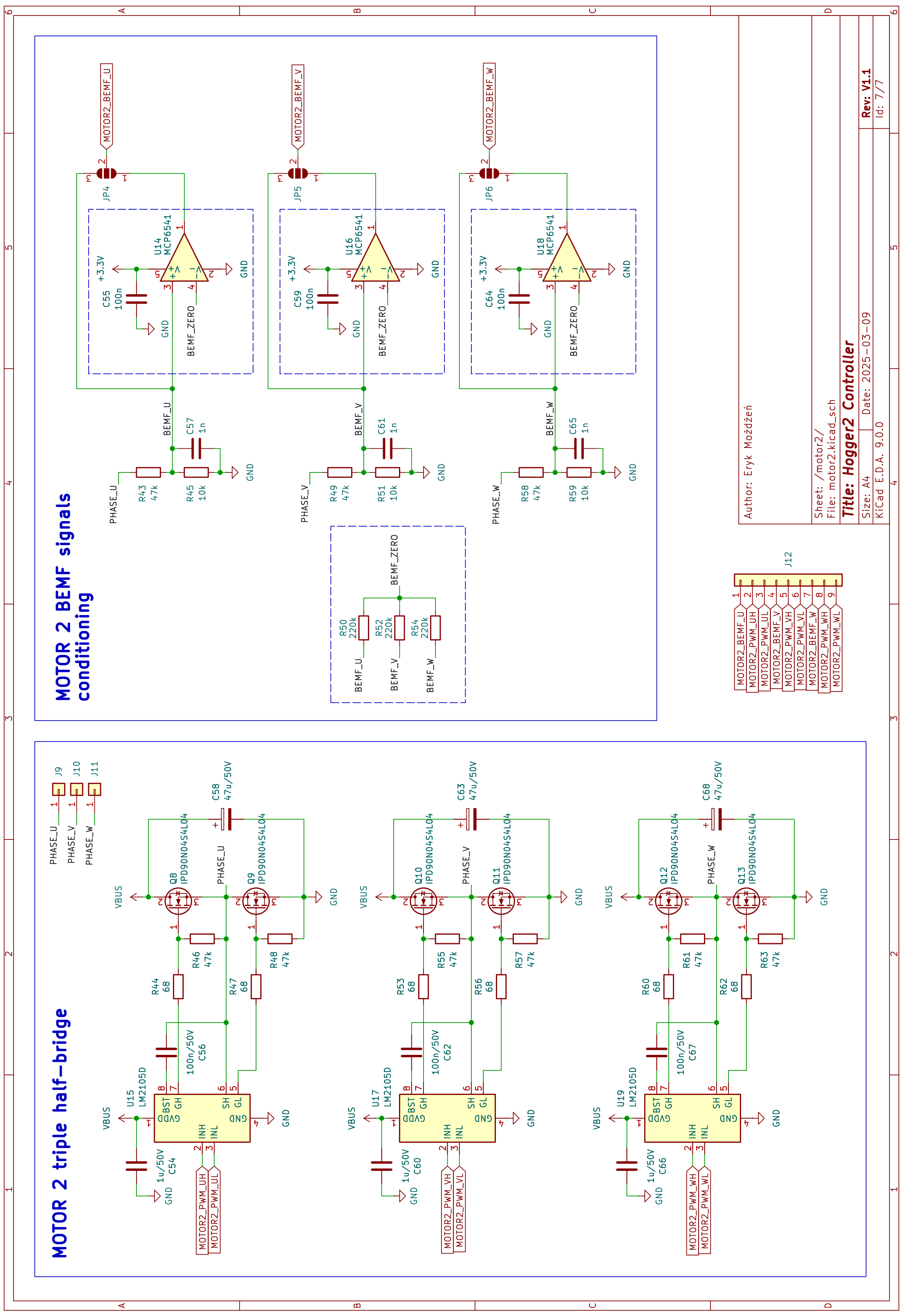


Figure A.6: Motor 2 control circuits

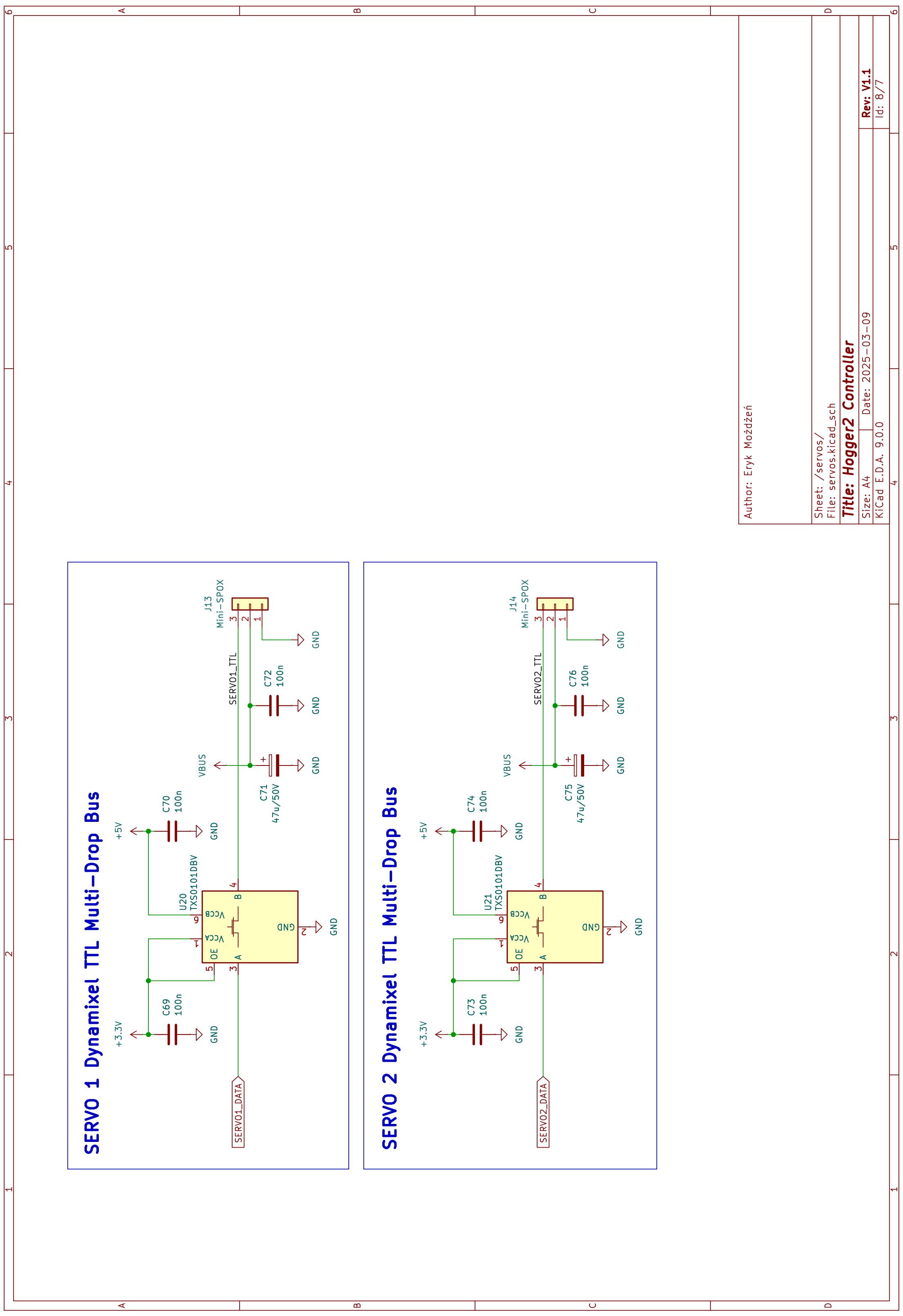


Figure A.7: Servomotor interfaces

Appendix B

Motor Controller Tunning

This appendix contains the results of motor step response analysis as well as PID controller parameters obtained from CHR 20% method [ÅH95]. The Table B.1 contains best-fit 4-parameter model values to the step responses collected from motors with wheels attached running in the air (no load conditions), geometry parameters and final controller settings. Data samples as well as tangent lines are shown in Figures B.1 and B.2.

Parameter		Motor 1	Motor 2
model	K_m	1260	1200
	T_{1m}	0.227	0.0614
	T_{2m}	0.227	0.396
	L_m	0.00870	0.0395
geometry	K	1260	1200
	T	0.616	0.557
	L	0.0726	0.0749
controller	K_p	0.00643	0.00590
	K_i	0.00745	0.00756
	K_d	0.000219	0.000208

Table B.1: Motors tunning parameters

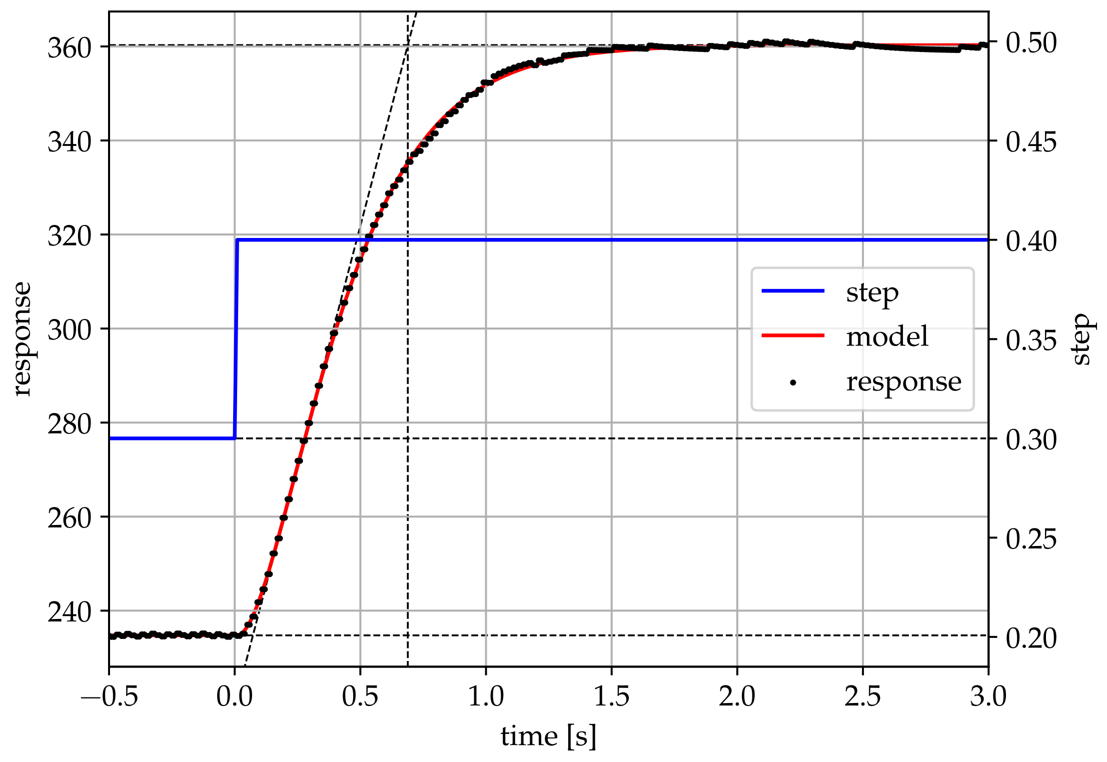


Figure B.1: Step response of motor 1

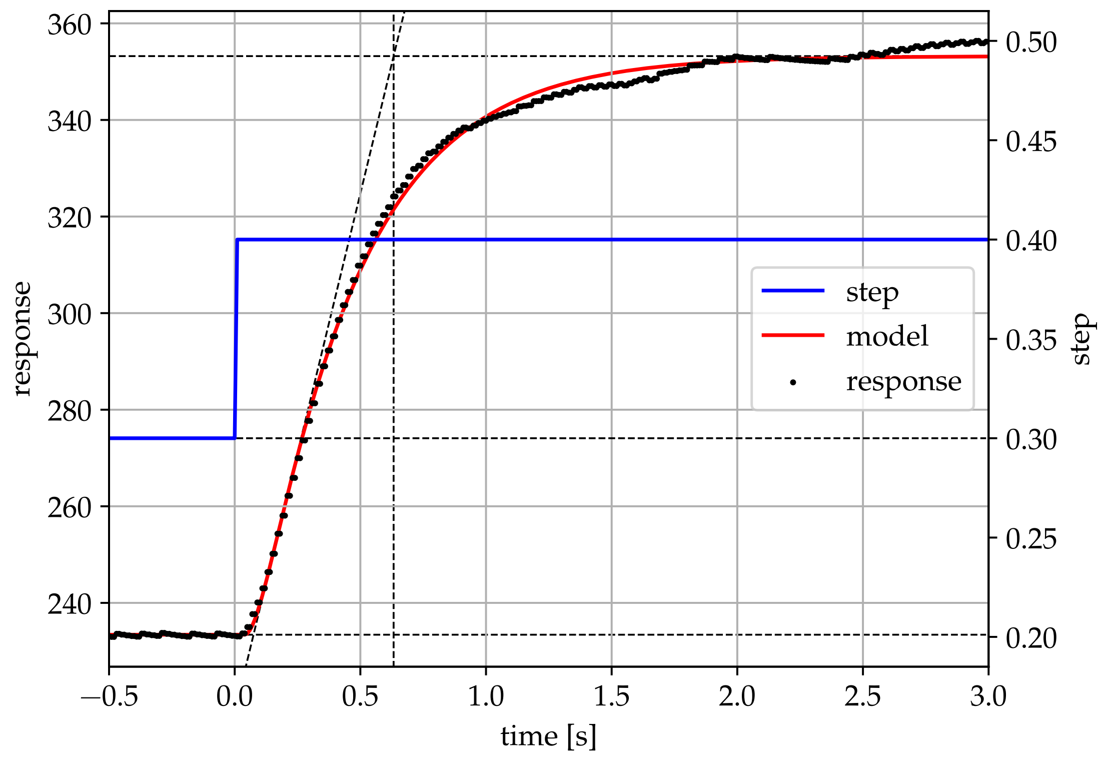


Figure B.2: Step response of motor 2

Appendix C

Cascade PID Tunning

This appendix presents the results of the robot's step response analysis, along with the PID controller parameters obtained using the CHR 0% overshoot method and the λ-tuning method [ÅH95]. The Table C.1 contains best-fit 4-parameter based on step responses of the robot's velocities during inner loop tuning, along with geometric parameters and the final inner controller settings. Table C.2 presents the geometric characteristics extracted from step responses of the robot's positions during outer loop tuning, together with the final outer controller settings. Step response plots for the inner system are shown in Figures C.1, C.2, and C.3, while the outer system responses are shown in Figures C.4, C.5, and C.6. All step response experiments were conducted with a fixed value of $\dot{\psi} = -250\ \frac{\text{rad}}{\text{s}}$.

Parameter		Controller x	Controller y	Controller θ
model	K_m	1.59	0.980	1.20
	T_{1m}	0.00136	0.147	0.114
	T_{2m}	0.514	0.147	0.295
	L_m	0.0296	0.133	0.0203
geometry	K	1.59	0.980	1.20
	T	0.522	0.399	0.537
	L	0.0426	0.175	0.0689
controller	K_p	3.73	0.816	2.26
	K_i	5.95	1.70	3.51

Table C.1: Inner loop tuning parameters

Parameter		Controller x	Controller y	Controller θ
geometry	K	0.905	0.932	1.04
	T	0.259	0.296	0.339
controller	λ	1	1	0.25
	K_p	4.26	3.63	11.3
	K_d	1.10	1.07	3.84

Table C.2: Outer loop tuning parameters

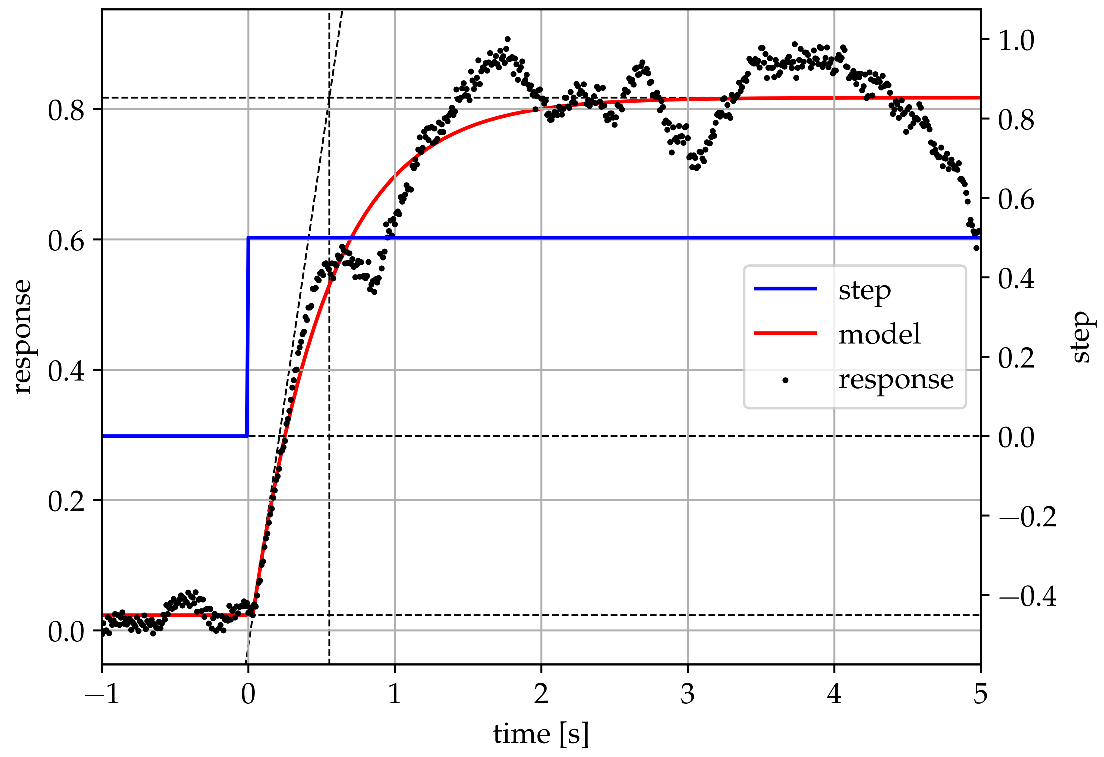


Figure C.1: Step response of inner system in x axis

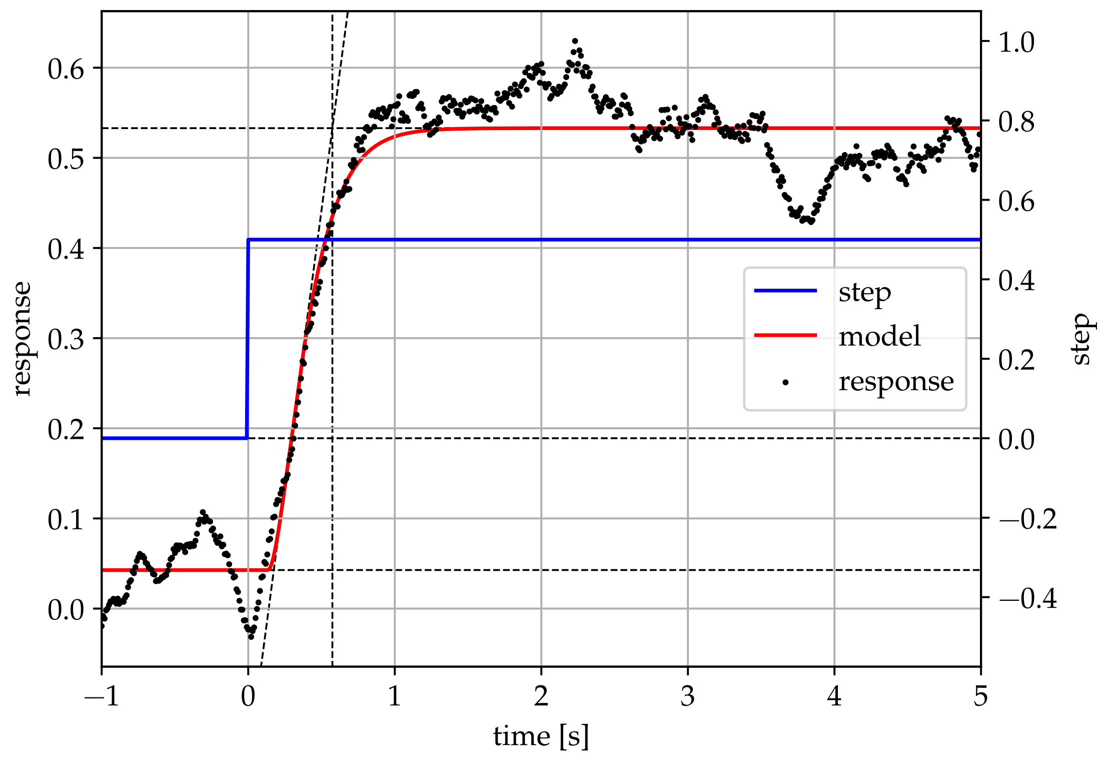


Figure C.2: Step response of inner system in y axis

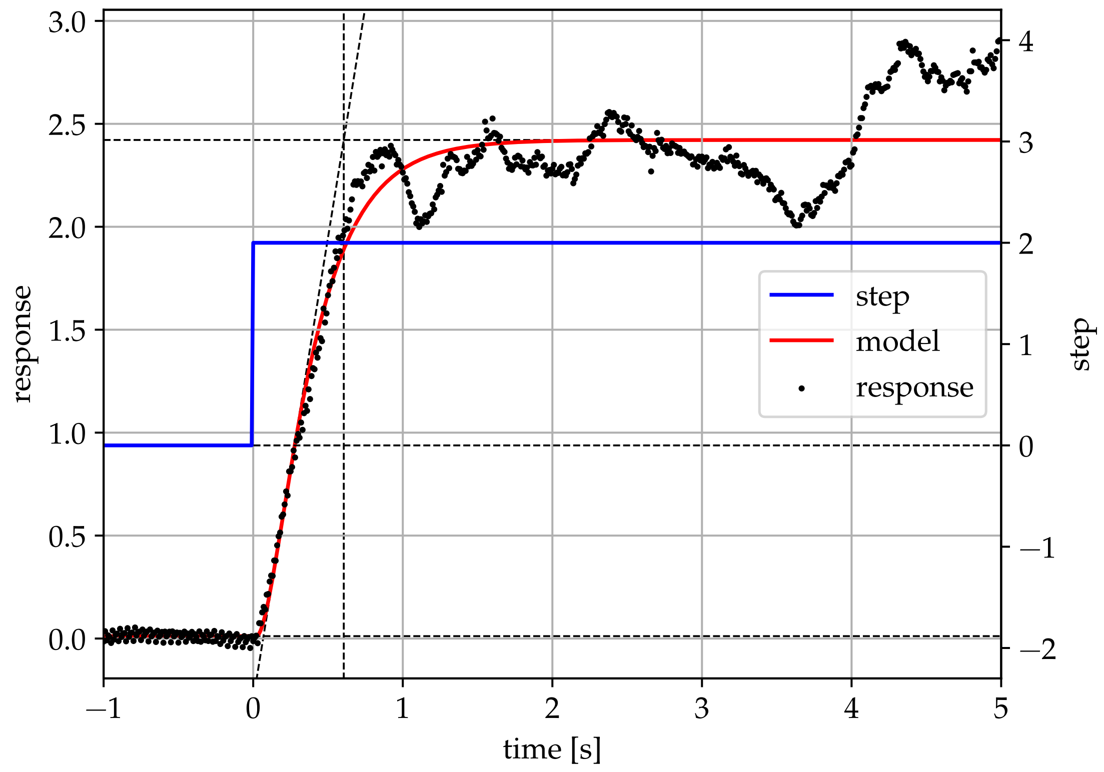


Figure C.3: Step response of inner system in θ axis

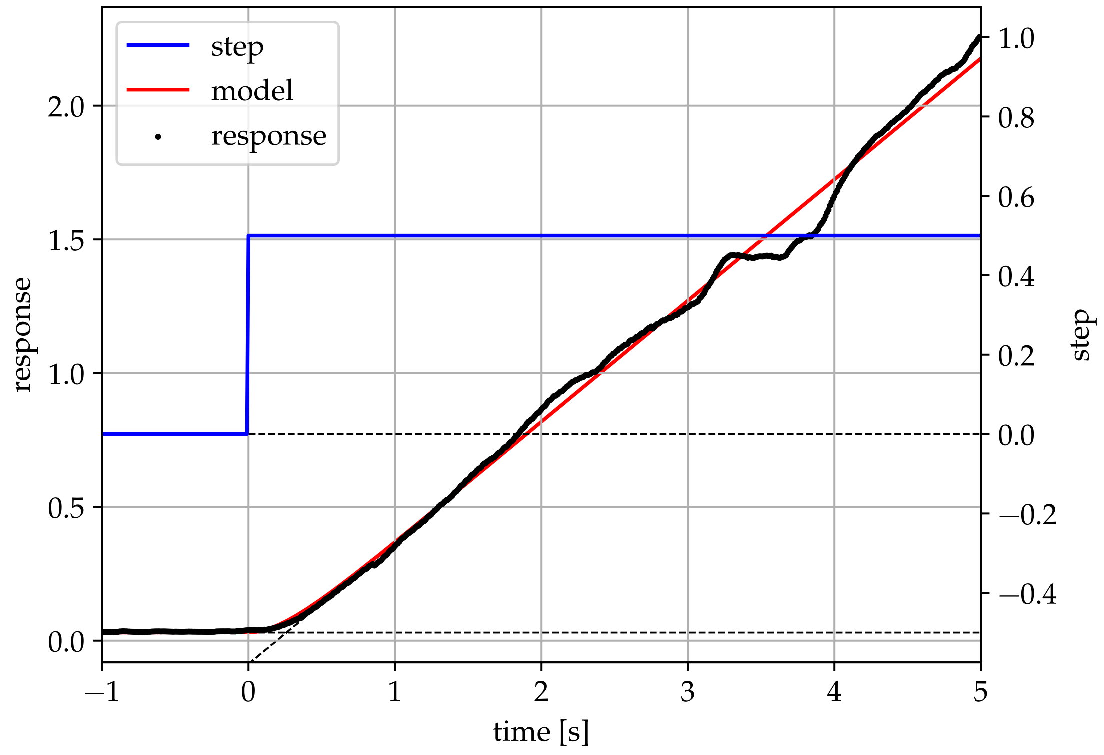


Figure C.4: Step response of outer system in x axis

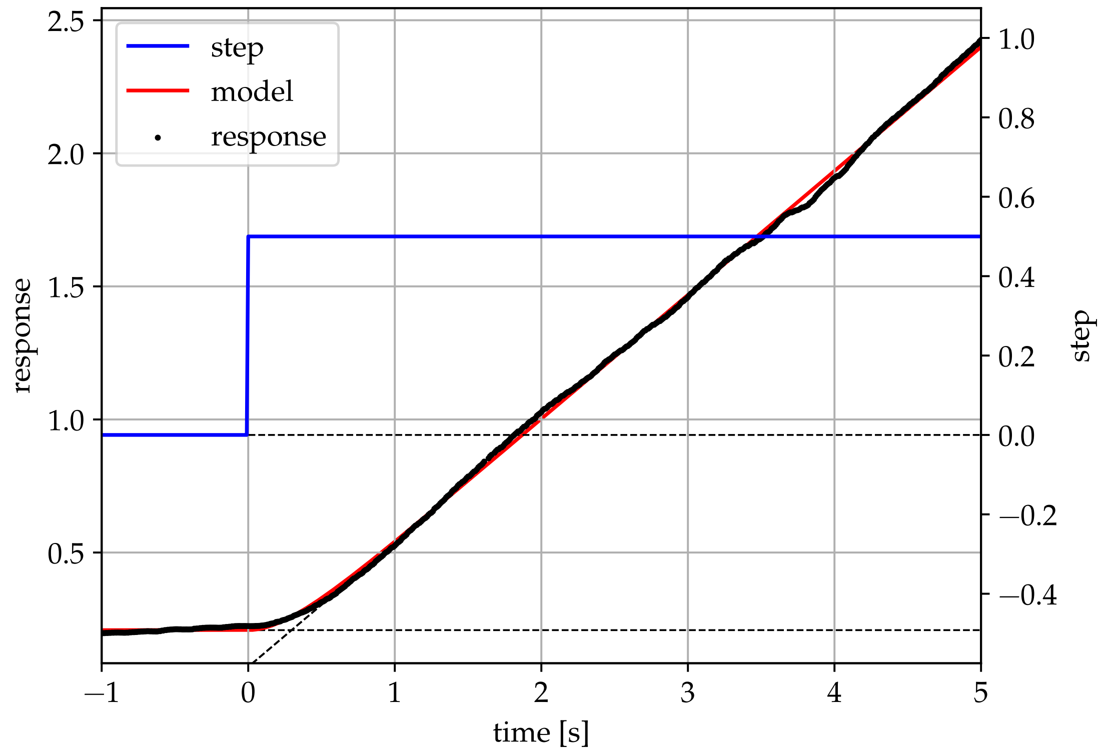


Figure C.5: Step response of outer system in y axis

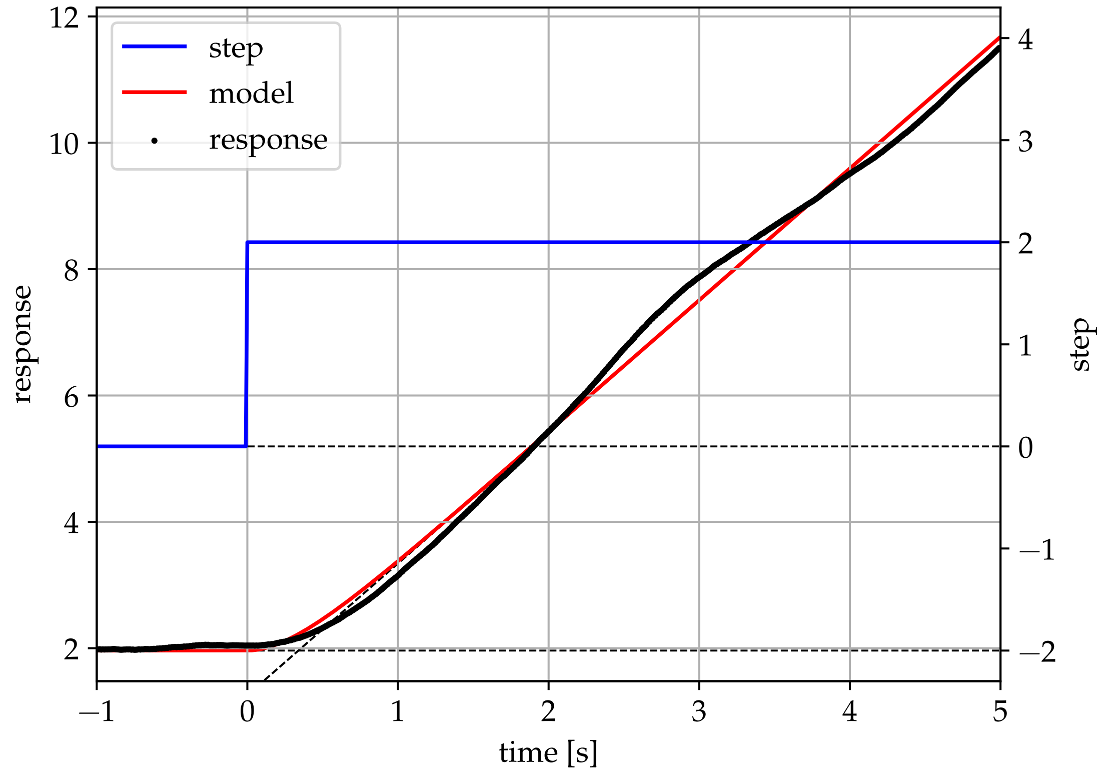


Figure C.6: Step response of outer system in θ axis