

ENERGY-EFFICIENT CONTROL STRATEGIES FOR WEARABLE EXOSKELETONS TO REDUCE FATIGUE DURING LONG-DISTANCE WALKING

Zhanbota Nigmatolla, B. Eng

**Submitted in fulfillment of the requirements for the degree of Master of
Science in Electrical and Computer Engineering**



**School of Engineering and Digital Sciences
Department of Electrical and Computer Engineering
Nazarbayev University**

53 Kabanbay Batyr Avenue,
Astana, Kazakhstan, 010000

Main supervisor: Prof. Prashant Jamwal
Co-Supervisor: Prof. Aibek Niyetkaliyev

April 2026

DECLARATION

I hereby declare that this manuscript, entitled “ENERGY-EFFICIENT CONTROL STRATEGIES FOR WEARABLE EXOSKELETONS TO REDUCE FATIGUE DURING LONG-DISTANCE WALKING”, is the result of my own work except for quotations and citations which have been duly acknowledged.

I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or international institution.



Name: Zhanbota Nigmatolla

Date: March 31, 2026

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Prof. Prashant Jamwal and Prof. Aibek Niyetkaliyev, for their invaluable guidance, expertise, and dedication throughout the duration of my Master's program. Their guidance, especially in the field of control theory and robotics, has played a key role in determining the nature of this work as well as its direction. The countless hours of guidance and interchange of ideas which they so generously offered have been priceless, and I have been indeed fortunate to be allowed to study under their supervision.

I also want to take this opportunity to extend my sincere appreciation to the administrative staff of Nazarbayev University whose tireless behind the scenes work underpins the academic environment within which this research was conducted. They have made contributions which though often stay unacknowledged, provide the much needed backbone on which the student community can achieve excellence.

Finally, my deepest gratitude to my family and friends, who have been a pillar of strength throughout this process. Particularly, I have to express my gratitude to Bright Egbo, Ykylas Alpanov, and Emmanuel Emeruom who became my companions and supported me throughout this program. I would especially like to mention the name of Bright Egbo, the cooperation with whom as a co-author on the publications in the top-tier journals became one of the most gratifying milestones of this academic journey.

Abstract

Gait and balance disorders affect approximately one-third of adults aged 60-97 years, with prevalence exceeding 60% in individuals over the age of 80, profoundly impairing mobility, independence, and quality of life. Powered hip exoskeletons offer a promising assistive solution; however, their transition to real-world, long-distance use is constrained by fixed-parameter controllers that cannot adapt to progressive, fatigue-induced changes in gait kinematics, causing systematic misalignment between robotic assistance and natural biomechanics.

This thesis presents the design, development, and preliminary simulation-based validation of an energy-efficient adaptive control architecture for a wearable hip exoskeleton targeting long-distance walking assistance. A lightweight bilateral prototype was constructed using three-dimensional (3D)-printed Polylactic Acid (PLA) and Acrylonitrile Butadiene (ABS) thermoplastics, actuated by CubeMars AK45-10 quasi-direct drive motors selected for their high torque density and mechanical back-drivability. A distributed dual-processor architecture was implemented: a Raspberry Pi 4 executes the high-level control logic, while an ODrive v3.6 motor controller manages low-level field-oriented control (FOC) and hardware safety constraints. Gait phase detection relies exclusively on the actuator's internal encoder, conditioned through a Low-Pass Filter (LPF) with a kinematic deadband to suppress control chattering.

The core control strategy combines a two-state Finite-State Machine (FSM) for real-time gait phase classification with a velocity-proportional admittance law for assistive torque generation. Validation through a 4-second Python simulation yielded a peak assistive torque of 5.0 Nm, a peak mechanical power of 11.6 W, and a cumulative energy injection of 2.38 J per stride. The FSM detected a swing phase fraction of 48.6% under the sinusoidal kinematic model used in the simulation. Compared to an unassisted baseline, the admittance strategy reduced surrogate joint effort demand by 47.7%, exceeding the 15% simulation target stated in the research proposal. Full metabolic validation through human trials and Human-in-the-Loop (HIL) optimization remains as future work.

Table of Contents

Acknowledgements	3
Abstract	4
List of Abbreviations & Symbols	8
List of Tables	10
List of Figures	11
List of Publications	12
Chapter 1 – Introduction.....	13
1.1 Background.....	13
1.2 Problem Statement.....	14
1.3 Aims and Objectives	15
1.3.1 General Objective:	15
1.3.2 Specific Objectives:	15
1.3.3 Contributions of this Thesis.....	16
1.4 Research Questions and Hypotheses	17
1.4.1 Research Questions:	18
1.4.2 Hypotheses:	18
1.5 Thesis Structure	19
Chapter 2 – Background and Literature Review	20
2.1 Survey of Deployed Robotic Hip Exoskeleton Systems	20
2.2 Mechanical Design and Actuation Modalities.....	22
2.3 Sensor Architectures for Human Kinematic Tracking.....	23
2.4 Hierarchical Control Strategies for Human-Robot Interaction.....	24
2.5 Human-in-the-Loop Optimization.....	25
2.6 Applications of Machine Learning and Reinforcement Learning	26
2.7 Research Gap.....	27
Chapter 3 – Methodology	29
3.1 Hardware Design	29
3.1.1 Frame Materials and Manufacturing	30
3.1.2 Design Requirements.....	30
3.1.3 Mechanical Architecture.....	32
3.1.4 Actuation System and Motor Selection	32
3.1.5 Cable-Driven Transmission	35
3.1.6 Thigh Linkage and User Interface	36

3.2 Control Architecture Overview	36
3.3 Gait Phase Detection (FSM).....	40
3.3.1 Signal Frequency Content and Filter Cutoff Rationale	42
3.3.2 Filter Order and Zero-Phase Implementation.....	42
3.3.3 Deadband Threshold Selection and Comparative Rationale	43
3.3.4 FSM State Transition Summary	43
3.4 Admittance Control Strategy	44
3.4.1 Transparent Mode (Stance Phase, State 0):	44
3.4.2 Assistive Mode (Swing Phase, State 1):	45
3.4.3 Derivation and rationale of ω_{max}	46
3.4.4 Saturation behavior and safe operating limits	46
3.4.5 Integration with Future Machine Learning Optimization:	46
3.4.6 Sensitivity of τ_{peak} and justification of the chosen value.....	47
Chapter 4 – Preliminary Results and Simulation	48
4.1 Kinematic and Energetic Simulation Results	48
4.1.1 Signal Processing and State Estimation:	49
4.1.2 Torque Generation and Biomechanical Energetics:.....	49
4.2 FSM Validation and Deadband Implementation	51
4.3 Biomechanical Energy Analysis	52
4.4 Hardware Status and Integration Plan	54
4.5 Baseline Comparison and Effort Reduction Analysis	56
4.6 Simulation Limitations	57
4.6.1 Constant stride frequency and amplitude.	57
4.6.2 Absence of gait asymmetry.....	58
4.6.3 No terrain variation or environmental disturbance.....	58
4.6.4 Absence of actuator dynamics.	58
4.6.5 Single-subject, no inter-subject variability.	59
Chapter 5 – Conclusion and Future Work	60
5.1 Conclusion.....	60
5.2 Future Work and Integration Roadmap	61
5.2.1 Phase 1: Hardware Assembly and Low-Level Calibration.....	61
5.2.2 Phase 2: Baseline Overground Human Trials.....	62
5.2.3 Phase 3: Machine Learning and Human-in-the-Loop Optimization	62
Bibliography.....	63

Appendix A	67
Appendix B	89
Appendix C	93

List of Abbreviations & Symbols

Abbreviations

3D	Three-Dimensional
AAN	Assist-As-Needed
ABS	Acrylonitrile Butadiene Styrene
API	Application Programming Interface
APO	Active Pelvis Orthosis
BLDC	Brushless Direct Current
BVS-HJE	Bionic Variable Stiffness Hip Joint Exoskeleton
CAD	Computer-Aided Design
CMA-ES	Covariance Matrix Adaptation Evolution Strategies
CoM	Center of Mass
CSJT	Cumulative Squared Joint Torque
DOF	Degree of Freedom
DSP	Digital Signal Processing
FDM	Fused Deposition Modeling
FOC	Field-Oriented Control
FSM	Finite-State Machine
GEMS(-H)	Gait-Enhancing and Motivating System (-Hip)
HIL	Human-in-the-Loop
HIL-RL	Human-in-the-loop – Reinforcement Learning
HWA	Honda Walking Assist
IMU(s)	Inertial Measurement Unit(s)
LPF	Low-Pass Filter
ML	Machine Learning
ML-HIL	Machine Learning – Human-in-the-loop
PLA	Polylactic Acid
PRA(s)	Pneumatic Rotary Actuator(s)
QDD	Quasi-Direct Drive
RL	Reinforcement Learning
RMS	Root Mean Square
ROM	Range of Motion
SEA(s)	Series Elastic Actuator(s)
sEMG	Surface Electromyography
USB	Universal Serial Bus

Symbols

A	Hip joint angle amplitude (degrees)
E	Cumulative mechanical energy injected (J)
$\varepsilon (\pm\varepsilon)$	FSM deadband half-width
f	Gait cycle frequency (Hz)
K_a	Admittance gain
$P(t)$	Instantaneous mechanical power (W)
t	Time (s)
δ	FSM deadband threshold velocity (rad/s)
$\theta(t)$	Hip joint angle as a function of time (rad or °)
$\tau^2(t)$	<i>Squared torque (in CSJT integral)</i>
$\tau_{cmd}(t)$	Commanded actuator torque (Nm)
τ_{peak}	Maximum allowable assistive torque (Nm)
$\omega(t)$	Hip joint angular velocity (rad/s)

ω_{max}

Normalization / saturation angular velocity
(rad/s)

List of Tables

Table 3.1 Hardware design requirements for the wearable hip exoskeleton prototype.....	31
Table 3.2 Key specifications of the CubeMars AK45-10 quasi-direct drive actuator	33
Table 4.1 Summary of simulation performance metrics for the baseline admittance controller...	51

List of Figures

Figure 3.1 CAD model of the bilateral wearable hip exoskeleton prototype, designed in Autodesk Fusion 360.	29
Figure 3.2 Assembled hardware prototype of the bilateral wearable hip exoskeleton.	34
Figure 3.3 Three-tier hierarchical control architecture of the wearable hip exoskeleton.....	37
Figure 3.4 Block diagram of the gait-phase-detection pipeline and its two-state FSM.	42
Figure 4.1 Simulation of the hierarchical control architecture..	49
Figure 4.2 Parameter sensitivity analysis illustrating the effect of varying the FSM deadband threshold and peak assistive torque on energetic output..	52
Figure 4.3 Biomechanical energy analysis of the exoskeleton-assisted gait cycle...	53
Figure 4.4 Bench-level bilateral closed-loop motor commissioning results....	56
Figure 4.5 Baseline versus assisted joint effort comparison derived from the sinusoidal kinematic model....	56

List of Publications

Nigmatolla, Z., Egbo, B., Riaz Khawaja, A., Shoaib Khan, M., Zhaisanbek, A., Niyetkaliyev, A., Kapsalyamov, A., Sagidoldin, D., and Jamwal, P. K., "A Systematic Review of Robotic Hip Exoskeletons for Gait Rehabilitation: From Design to Clinical Impact," *Journal of NeuroEngineering and Rehabilitation*, submitted March 2026. (*Under Review*)

Egbo, B., **Nigmatolla, Z.**, Khan, N. A., and Jamwal, P. K., 2025, "Explainable Machine Learning for Early Detection of Parkinson's Disease in Aging Populations Using Vocal Biomarkers," *Frontiers in Aging Neuroscience*, 17, p. 1672971. doi: 10.3389/fnagi.2025.1672971

Chapter 1 – Introduction

1.1 Background

Gait and balance disorders are a significant and growing public health problem, mainly attributed to the process of aging in the global population structure [2]. Epidemiological studies have shown that one-third of adults in the 60- to 97-year age range experience gait disorders, with a prevalence rate of more than 60% in people older than 80 years of age [2]. Although orthopedic and sensory factors contribute significantly to gait and balance disorders, neurological causes are responsible for almost one-quarter of the elderly population, and stroke is a major cause of long-term ambulatory disability in people worldwide [3].

In terms of biomechanics, the gait dysfunction resulting from neurological or musculoskeletal degenerations is associated with severe impairments in coordination, balance, and force generation [4, 5]. More specifically, the weakening of the hip flexors, extensors, and abductors is known to directly affect the stride, propulsion, and pelvis during gait [6, 7]. This not only increases the risk of falls but also reduces the energetic efficiency of the gait, resulting in the quick onset of fatigue and the drastic reduction in the quality of life of the individual [4, 8].

To address these major biomechanical deficiencies, wearable robotic exoskeletons have been proposed as a highly promising technology for assisting individuals with mobility impairments [6]. Exoskeletons are wearable mechatronic devices intended to boost human physical capacities by mechanically assisting human limbs [5, 6]. Although these technologies were originally created to help soldiers carry loads, they have successfully been adapted to help post-stroke patients, the elderly, and industrial workers [5, 9]. In fact, there has been increasing interest in powered hip exoskeletons because the hip joint plays an important role in regulating the control of the center of mass (CoM) and contributes significantly to the mechanical power output during walking on level ground [1, 8].

Although these devices show promise in the clinical setting, there is a major engineering barrier to the widespread adoption of these devices into the real world [5, 8, 10]. The first and main issue is the efficiency of the devices, which affects both the battery life of the machine and the metabolic cost to the user [11]. During long-distance walking, the human neuromuscular system naturally adapts to terrain changes and cumulative exhaustion [12, 13]. If the exoskeleton is unable to adapt to the changing physiological state, it may actually work against the natural biomechanics of the user, thereby increasing the metabolic cost of the walk instead of decreasing it [14, 15].

Recent advances in Artificial Intelligence and Machine Learning (ML) have provided a promising solution for overcoming this limitation [16, 17]. The integration of high-resolution wearable sensors with lightweight microcomputers allows modern exoskeletons to process complex, high-dimensional kinematic data in real-time [18]. By moving away from static, pre-defined assistance strategies to more adaptive, ML-based control paradigms, it has been shown that exoskeletons can potentially provide customized assistance profiles to match the exact demands of the user [19, 20]. Thus, the development of intelligent control strategies that optimize electrical and human metabolic efficiency is essential for the establishment of exoskeletons as viable solutions for the aging or impaired population [4, 21].

1.2 Problem Statement

Despite the tremendous progress achieved in the hardware and actuation aspects of exoskeletons, the control strategies adopted are still inefficient in terms of energy expenditure, thereby limiting the practical viability of the system for long-distance walking [4, 10]. The majority of the existing commercial and state-of-the-art exoskeletons are based on rigid and model-based control strategies, namely fixed parameter impedance control and admittance control, which are based on fixed levels of assisting torque [5, 6]. Although the static control strategies are appropriate for short and highly controlled rehabilitation sessions, they are critically deficient for practical purposes. The fixed levels of torque are unable to cope with inter-subject variability (unique walking patterns of each user) and intra-subject variability (dynamic accumulation of fatigue and changing environmental conditions) [14, 19].

This lack of flexibility causes a harmful disconnect between the pre-programmed actuation timing of the device and the changing gait biomechanics of the user [14, 15]. As a user walks over a large distance, muscle fatigue causes changes in stride length, swing rate, and overall gait rhythm [22]. While the fixed controller blindly continues to apply torques based on its pre-programmed actuation timing, it causes a disconnect between the user and the exoskeleton. Rather than providing assistance, it causes resistance to the user's natural biomechanics. This disconnect directly causes a waste of electrical power supplied by its battery and, more importantly, causes increased metabolic costs on the human user. Indeed, it causes increased rates of fatigue for the user and drastically decreases the range of the exoskeleton [8, 11].

Though the traditional Human-in-the-Loop (HIL) optimization approach has been successfully utilized to minimize the amount of energy wasted in the past, it has been observed to demand high convergence times, which are not suitable in dynamically changing real-world scenarios [14, 20, 23]. Reinforcement Learning (RL), an approach within the domain of Machine Learning, has been

identified as a potentially fruitful area of research in the near future [16, 19]. Yet, the seamless integration of the RL approach with efficient, real-time sensor processing pipelines, which can filter the noisy kinematic data without demanding high latencies, is highly needed [16, 18].

The main problem addressed in this research is the lack of an adaptive and energy-efficient control system capable of real-time personalization. There is an urgent need to develop a hybrid Machine Learning – Human-in-the-Loop (ML-HIL) control system that can not only effectively identify the gait phase using efficient low-level logic but can also dynamically regulate the torques to reduce human fatigue and electric energy consumption over long walking distances [4, 19].

1.3 Aims and Objectives

1.3.1 General Objective:

The main aim of this study is to develop and validate adaptive and energy-efficient control strategies for a wearable hip exoskeleton for extended periods of user walking assistance. By utilizing real-time information from user motion sensors and a hybrid approach combining machine learning and HIL optimization, it is possible to optimize assistive torques in real-time for individual users. The overall aim is to reduce the metabolic cost for the user, by at least 15% relative to unassisted walking, while increasing the electrical efficiency of the actuators.

1.3.2 Specific Objectives:

To achieve the general objective, the research is divided into the following specific, measurable goals:

1. **Review and Compare Existing Architectures:** To undertake an extensive analysis of the existing control strategies adopted in the control of exoskeletons, namely, rigid impedance control, traditional HIL optimization, and the newly developed control strategies using machine learning, in order to identify the major limitations in the adaptability and energy efficiency of the system.
2. **Develop a Hybrid ML-HIL Control Algorithm:** To develop a hierarchical control system consisting of an accurate Finite-State Machine (FSM) control system for accurate gait phase recognition and the newly developed control system using the machine learning algorithm to adapt the peak assistive torque in real time.
3. **Prototype and Simulate the System:** To develop an accurate and efficient hip exoskeleton system using 3D printing techniques with the ability to use quasi-direct drive actuators, and undertake detailed mathematical modeling and kinematic simulations to test the control system.

4. **Evaluate Energetic and Biomechanical Performance:** To test the system and assess its efficiency by calculating the mechanical energy contributions, the reduction in metabolic cost, the improvement in the stability of the gait, and the improvement in the battery life while walking over long distances.

1.3.3 Contributions of this Thesis

The objectives stated in Section 1.3.2 define the overall arc of the research program, of which the human-subject evaluation and the machine-learning-based personalization components remain future work as outlined in Chapter 5. This subsection enumerates the concrete contributions delivered within the present thesis, so that the scope of the claims made in Chapters 3 and 4 is established in advance of the methodology and results.

The contributions of this thesis are as follows:

- **Mechanical design of a bilateral wearable hip exoskeleton prototype.** A lightweight bilateral hip exoskeleton was designed in Autodesk Fusion 360 and fabricated using PLA and ABS thermoplastics via fused deposition modelling, with a 3D-printed pelvic frame, lateral rigid uprights, and thigh-cuff interfaces. Actuation at each hip joint is delivered by a CubeMars AK45-10 quasi-direct-drive motor coupled through a custom double-groove cable-driven transmission. The detailed design, including the hardware requirements table and the mechanical architecture, is presented in Section 3.1 and Figure 3.1.
- **Distributed two-tier control architecture.** A hierarchical control architecture was implemented that separates the cognitive and execution layers onto physically distinct computing devices: a Raspberry Pi 4 running the Python high-level control pipeline at 500 Hz, and an ODrive v3.6 motor controller handling low-level field-oriented current control, hardware-enforced current, velocity, and thermal limits, and a 500 ms hardware watchdog. The bypass of the CubeMars AK45-10 internal driver circuitry and the use of a USB-based ODrive Python API interface between the two tiers are design decisions made and justified in this thesis. The complete architecture is presented in Section 3.2.
- **Encoder-only gait-phase detection pipeline.** A signal-processing and phase-detection pipeline was designed that derives the gait phase exclusively from the AK45-10 internal magnetic encoder, without external inertial measurement units or foot-mounted force sensors. The pipeline combines a 4th-order zero-phase Butterworth low-pass filter at 5 Hz with numerical differentiation and a two-state finite-state machine that distinguishes stance from swing using a single velocity deadband of 0.1 rad/s. The selection of the filter order

and cutoff, and the derivation of the deadband threshold from the post-filter noise floor of the velocity signal, are justified in Sections 3.3.1 through 3.3.3.

- **Velocity-proportional admittance control law with analytically derived saturation.** An admittance control law was formulated that commands a velocity-proportional assistive torque during the swing phase and zero torque during the stance phase. The normalization parameter $\omega_{\max}$ in the admittance equation is derived analytically from the kinematic model of the user-exoskeleton system, yielding a peak angular velocity of approximately 2.19 rad/s at $A = 20^\circ$ and $f = 1.0$ Hz. The saturation behavior at the chosen peak torque of 5.0 Nm is reconciled against the AK45-10 continuous and peak torque ratings. The complete derivation is presented in Section 3.4.
- **Custom Python simulation framework and energetic validation.** A simulation framework was developed in Python that implements the complete DSP, FSM, and admittance pipeline, together with extensions for per-stride energy statistics, a torque sensitivity sweep, a bilateral symmetry check, and a cumulative squared joint torque comparison between an unassisted baseline and the exoskeleton-assisted condition. The simulation produces the quantitative metrics reported in Chapter 4: a peak assistive torque of 5.0 Nm, a mean swing-phase torque of 2.95 Nm, a stride-averaged injected energy of 2.38 J, a peak mechanical power of 11.6 W, a swing phase fraction of 48.6%, and a 47.7% reduction in the surrogate CSJT effort metric compared with the unassisted baseline. The CSJT reduction is a simulation-based surrogate measure of effort and is not a measurement of metabolic cost; this distinction is preserved throughout Chapter 4 and addressed in Section 4.6.

The contributions listed above constitute the deliverables of the present work. Items that remain outside the scope of this thesis, including the physical commissioning of the hardware prototype, the integration of a machine-learning-based or Bayesian-optimization-based personalization layer, and human-subject evaluation of the metabolic benefit, are proposed as future work in Chapter 5.

1.4 Research Questions and Hypotheses

By guiding the creation and assessment of an energy-efficient control system, this thesis research is aimed to address several questions through crucial issues concerning machine learning, biomechanics, and wearable robotics.

1.4.1 Research Questions:

1. **Efficacy of ML-Based Control:** How effective are adaptive machine learning-based control systems when compared to traditional fixed-parameter control systems and standard HIL optimization techniques in terms of reducing the metabolic cost of walking over an extended duration?
2. **Real-Time Personalization:** Which sensor integration techniques and machine learning systems can be utilized effectively for fast and real-time personalization of an exoskeleton's assistance in accordance with changes in an individual user's gait and fatigue?
3. **Fatigue Reduction in Varied Conditions:** How effectively can a hybrid ML-HIL control system reduce different indicators of fatigue, such as muscle activity and kinematics, under different conditions of walking, such as terrain and velocity?
4. **Safety and Stability:** How can user safety and biomechanical stability be assured within an adaptive hybrid control system, especially during different gait phase changes and unexpected user movements?

1.4.2 Hypotheses:

On the basis of the current literature, the hypotheses to be tested in the current research, according to the proposed methodology, are the following:

- **Hypothesis 1 (Metabolic Reduction):** The results of the simulation are in line with the expectations, confirming the hypothesis that the proposed architecture of the admittance control will reduce the surrogate joint effort demand, which will be over 15%. Future research will focus on the validation of the results via the optimization of the ML-HIL and human subjects.
- **Hypothesis 2 (Gait Stability):** The proposed control algorithm will enhance the stability of the gait of the user, as the assistive torque will be precisely coordinated according to the gait phases of the user, which will lead to a 15-25% reduction in kinematic variations from a stable, nominal gait pattern.
- **Hypothesis 3 (System Efficiency):** The proposed personalized control algorithms will enable the exoskeleton to operate for a longer period, optimizing the delivery of the torque, which will lead to the minimization of the electric energy consumption compared to the traditional control approach.

1.5 Thesis Structure

This research thesis comprises five chapters, which describe the clinical background, engineering approach, and preliminary validation of the proposed energy-efficient approach in designing an exoskeleton controller.

- **Chapter 1: Introduction** describes the clinical and engineering background associated with gait disorders, defines the problem associated with energy inefficiency, and explains the research hypothesis and aims.
- **Chapter 2: Literature Review** provides a detailed overview of existing robotic hip exoskeletons, including their mechanical design, actuation mechanism, and clinical implications. The chapter concludes with an overview of the research gap in existing literature.
- **Chapter 3: Methodology** describes the engineering development process for the proposed system. This includes the mechanical Computer-Aided Design (CAD) development of the proposed lightweight hip exoskeleton, distributed control using ODrive, digital signal processing (DSP) for FSM-based gait phase detection, and the mathematical derivation of the proposed admittance control strategy.
- **Chapter 4: Preliminary Results and Simulation** discusses the validation process for the proposed control architecture. This includes Python-based high-fidelity simulations demonstrating the ability to effectively filter sensor noise, detect gait phase, and inject positive mechanical work into the human gait cycle. A parameter sensitivity study that determines the impact on energetic output of varying peak assistive torque and FSM-based deadband thresholds are also included. Bilateral control is also simulated, demonstrating symmetric control output for both left and right actuation channels.
- **Chapter 5: Conclusion and Future Work** discuss the achievements of the current development phase and describes what steps will be implemented in the future. This includes porting the validated simulation logic onto physical hardware and integrating the RL optimizer to minimize metabolic cost during long-distance walking.

Chapter 2 – Background and Literature Review

2.1 Survey of Deployed Robotic Hip Exoskeleton Systems

In the past decade there has been a development of a large variety of hip exoskeleton platforms, many of which include laboratory-based research exoskeletons tethered to a stationary laboratory and commercially produced exoskeletons for clinical rehabilitation and community use. Unlike in Chapter 1, where the epidemiological and biomechanical motivation for these devices was presented, this section will focus on a review of literature with respect to design niche of the exoskeleton described in this thesis. The comparison establishes the particular design niche of the exoskeleton described in this thesis.

One of the most popular commercially available systems is the Samsung Gait-Enhancing and Motivating System (GEMS) and its clinical version the Samsung GEMS-H. This system uses a lightweight hip bilateral with brushless direct current (BLDC) motors and adaptive oscillator-based gait detection and provides gait enhancement in the context of age-related gait disorders in both community and clinical settings [24]. The assist mode has been utilized in research with chronic stroke patients, showing positive results in terms of walking speed and stride length in stroke rehabilitation patients [25, 26]. Another similar exoskeleton, the Bot Fit, is a bilateral walking exercise assist robot developed by Samsung, which has been tested in RCTs with older adults, showing positive results in terms of walking speed and TUG efficiency with various levels of assistance [27]. The common drawback for both assisting systems is their inability to continuously adjust their torque profiles according to the progressive biomechanical changes occurring during long-distance walking.

The Active Pelvis Orthosis (APO), by BioRobotics Institute at Scuola Superiore Sant'Anna, is one of the best documented and validated systems in the peer-reviewed literature [28]. The system offers assistance to the hip flexion and extension movements bilaterally in the sagittal plane and was validated for individuals with chronic stroke and transfemoral amputation. The use of bilateral assistance with the APO was shown to improve walking speed by 0.14 m/s, on average, for individuals with impaired gait due to stroke, surpassing the minimal clinically important difference [29]. Another study showed that walking energy expenditure was reduced for individuals with above-knee amputations while walking over ground and assisted with the APO [30]. Mechanically, the system is designed to allow for safe force interaction, although its assistance is limited to the sagittal plane, which may be a concern for individuals with impaired stability in the frontal plane as well.

The commercial device, “Honda Walking Assist” (HWA), is designed to provide hip flexion and extension and has been used primarily among the elderly and postsurgical populations. Koseki *et al.* found a significant improvement in self-selected walking speed and step length among total knee replacement patients using HWA [31]. Firouzi *et al.* found improvements in spatiotemporal parameters among stroke survivors using the HWA device [32]. The parameters are set individually before each session, but they are fixed and static during the walking process, and this structural limitation makes the device unable to track the in-session biomechanical drift caused by accumulating fatigue of the session.

In the field of research exoskeletons, the Bionic Variable Stiffness Hip Joint Exoskeleton (BVS-HJE) proposed by Yang *et al.* features a cable-driven antagonistic variable stiffness mechanism that enables the device’s joint impedance to be adjusted independently of its output torque [33]. The variable stiffness feature has mechanical relevance in fatigue-adaptive control, as it would potentially allow the device's interaction stiffness to be reduced in accordance with a user's weakening gait rhythm. The BVS-HJE has only been subject to laboratory trials in its early stages of clinical validation [33].

Apart from rigidly framed design of the exoskeletons, soft robotic platforms have also shown that significant metabolic reductions are possible without the need for a rigid skeletal system. Ding *et al.* used a HIL approach with a Bayesian optimization strategy to optimize the peak and timing offset for hip extension assistance provided by a soft exosuit, and an average net metabolic cost reduction of $17.4 \pm 3.2\%$ was achieved compared to unassisted walking, with a convergence time of just 21.4 ± 1.0 minutes [14]. This is the most significant metabolic cost reductions for hip-specific assistance and also validated the use of a Bayesian optimization strategy for a HIL approach for wearable robotics. The soft exosuit was tethered and required an off-board actuation system, and the validation was performed only for steady-state walking on a treadmill at a constant speed of 1.25 m/s, without any validation for real-world scenarios involving varying speeds and long distances.

The new research prototypes have started incorporating hierarchical learning-based strategies from the control theory side directly into hip exoskeleton platforms. Zhang *et al.* presented a bilateral hip exoskeleton device with a FSM-based impedance controller implemented in a low level and a RL-based agent implemented in a high level, showing the ability of such a system to personalize hip joint assistance to improve range of motion for a variety of walking tasks in both unimpaired and one post-stroke participant [19]. A similar study from the same research team implemented a delayed output feedback controller and a least-squares policy iteration-based RL approach to improve gait symmetry, reducing an artificially introduced asymmetry index from 8.8% to -0.5%

in less than four minutes of online optimization time [7]. This shows the ability to implement RL-based personalization in physical hardware, although both studies were performed on a treadmill platform with fixed walking speeds.

Throughout the review it is evident that a pervasive and unsolved trade-off is present. The clinically accessible systems like GEMS, HWA, and APO have been realized with static or session-level torque control approaches that cannot account for within-session fatigue changes. Personalizing and metabolically improving systems with adaptive control have been realized for research purposes but have been validated only under steady-state conditions on a treadmill. There is no system reviewed here that can provide a lightweight quasi-direct drive (QDD) actuation, a fatigue-adaptive control, and an open hardware infrastructure for ML-HIL integration for overground and long-distance walking. These three aspects form the design goals of the proposed prototype, with the QDD actuation realized in hardware, the fatigue-adaptive control validated in simulation, and the ML-HIL integration planned as future work.

2.2 Mechanical Design and Actuation Modalities

The mechanical architecture of a wearable hip exoskeleton is a foundation that primarily determines the torque fidelity, mass, and suitability of the exoskeleton for a particular population group. According to the literature, the configurations of the exoskeletons are generally divided into two categories: unilateral and bilateral exoskeletons [6, 9]. The most studied exoskeletons in the current literature are the bilateral exoskeletons, which provide symmetrical torque support to the hip joints, thereby providing support to generalized weakness in the lower extremities and gait in the elderly [4, 6]. In contrast, unilateral exoskeletons are generally designed to support those with highly asymmetrical weakness, including those with hemiparesis due to stroke or those who are leg amputees [25, 29, 30].

Another very important aspect in the design of the wearable robots is the trade-off between power density and mechanical compliance. Currently, the electric motor-based robots, specifically the BLDC motor technology and the QDD actuators, are the most dominant in the field due to the highly compact nature of these technologies, coupled with the high precision in torque tracking [34]. To improve the safety and comfort of the human-robot interface, several researchers in the field have attempted to incorporate mechanical compliance in the drivetrain of the robots. Specifically, the Series Elastic Actuators (SEA) and the Pneumatic Rotary Actuators (PRA) enable the robots to safely deviate from the rigid paths, thereby allowing the incorporation of physical elasticity in the drive train of the robots, specifically between the motor and the human [9, 34]. However, the incorporation of mechanical compliance of the robots has been noted to compromise

the weight of the robots [34]. Consequently, the modern robots, including the one in the proposed research in this thesis, are increasingly incorporating the highly back-drivable QDD motor technology coupled with software-based virtual compliance, specifically admittance control, to ensure the robots are lightweight [34].

The mechanical aspects of the hip exoskeleton have also been explored through the development of devices that integrate multiple degrees of freedom (DOF) to counteract instability in the sagittal and frontal planes. Zhang *et al.* designed a bilateral hip exoskeleton with two DOFs, namely, the flexion-extension and abduction-adduction DOFs, each driven by a compact SEA consisting of a high-torque BLDC motor, 100:1 harmonic drive gearbox, and a custom-designed double-spiral titanium torsional spring [35]. The SEA actuators are designed to produce a continuous output torque of 40 Nm and a peak torque of 80 Nm, while maintaining a torque-to-weight ratio comparable to smaller and lower torque QDD actuators [35]. The inclusion of physical spring elements provides shock tolerance and passive compliance at the human-robot interface, at the cost of increased per-joint mass of approximately 1.5 kg and reduced control bandwidth relative to direct-drive alternatives [35]. This is another issue that arises when designing hip exoskeletons, with SEA-based designs having the advantage of inherent mechanical safety and accurate force control at the cost of higher mass and bandwidth, compared to QDD-based designs that have the advantage of lower mass and higher backdrivability at the cost of software-based admittance control to ensure safe and transparent human-robot interaction.

2.3 Sensor Architectures for Human Kinematic Tracking

For the exoskeleton to offer transparent and efficient assistance, the system must be able to detect the user's gait phase and infer the user's movement intentions in real-time [4, 6]. The literature shows the wide variety of sensor configurations that have been developed to attain this synchronization [5, 10]. The most fundamental method is the utilization of high-resolution joint encoder sensors, which are usually incorporated into the actuators themselves, permitting continuous measurements of the degree of flexion and extension of the hips [10, 18].

To ensure better adaptability towards the external environment, along with the ability to sense the entire human movement, devices have incorporated Inertial Measurement Units (IMUs) placed on the pelvis and/or thigh, which help in tracking the orientation and timing of strides taken during walking [10, 18]. Though better human-robot interaction schemes have incorporated multi-axis force and torque sensors, which help in measuring interaction force, such components have been shown to increase the cost and vulnerability of the system significantly [4, 6]. Moreover, though Surface Electromyography (sEMG) has been utilized in the lab to verify the reduction in muscle

activation, the system has been confronted with considerable challenges, making it difficult to apply this system, considering factors such as signal noise, sweat, and movement [4, 16]. Hence, developing better control schemes that rely on reliable and low-noise measurements from encoders and IMUs is a significant task in developing such systems [4, 10].

For the case of encoder and IMU-based gait phase estimation, two distinct methods have been developed, namely threshold-based rule methods and machine learning methods. The threshold-based rule methods make use of heuristic rules to identify the gait phase using the kinematic data, namely the angular velocities or accelerations, and detect the occurrence of events such as heel strike and toe-off. In the case of the hip, the cyclic nature of the angular velocities is appropriate for adaptive oscillators, in which the phase is determined by synchronizing an adaptive parametric oscillator to the measured gait frequency and using the phase estimate to control the torque application [4]. Although the threshold-based methods are computationally inexpensive, they are sensitive to the quality of the signal and are less accurate when the walking speed is suddenly changed or the position of the sensors is altered over an extended period [18]. The application of the machine learning methods overcomes the limitations of the threshold-based rule methods, and the mapping from the sensor data to the gait phase is learned using the data collected over the changing conditions. Gao *et al.* have developed a hybrid model consisting of LSTM, ResMLP, and LightGBM and have tested the model on an exoskeleton designed to carry loads up to 40 kg and moving over five modes, namely walking on the ground, walking on the stairs, and walking on the ramps, and have achieved an average early detection advancement of 63.79 ms compared to the ground truth gait events [18]. The early detection is significant in the case of the control of the exoskeleton, as the torque application is initiated before the occurrence of the mechanical event, and the phase lag is reduced between the biomechanical action and the action initiated by the exoskeleton. However, the application of the deep learning model is associated with much higher computational complexity compared to the threshold-based methods, and the implementation on the Raspberry Pi 4 requires careful attention to the latency and memory [16, 18]. In the case of the baseline model developed in the thesis, the encoder-based velocity threshold method is adopted because of its determinism, latency, and lack of need for the training data, and the application of the ML method is considered as the future direction.

2.4 Hierarchical Control Strategies for Human-Robot Interaction

The inherent intelligence in the robotic exoskeleton lies in the control strategy, which dictates the interaction of the exoskeleton with the human voluntary control input. Control strategies are generally categorized based on the level of intervention in the literature [6, 9, 10]. Trajectory-Tracking (Position) control has been extensively used in early rehabilitation exoskeletons. This

approach is very rigid in nature, dictating the user to move in a clinically normative kinematic trajectory [9]. Such an approach, while beneficial to completely paralyzed users, has been shown to impede the user in cases of residual muscle control, actually working against neuroplasticity and increasing metabolic costs as the user struggles against the exoskeleton [4, 10].

To overcome this, modern research efforts have focused on the implementation of "compliant" strategies, most prominently the Admittance Control and Assist-As-Needed (AAN) strategies [4, 10]. Admittance control does not impose an intended path; instead, the user's kinematic input, such as angular velocities, is used to compute the generated torque output [34]. The generated output is essentially an "virtual spring" or "virtual damper" that enhances the user's movements while maintaining high levels of transparency. The user is thus free to control the walking speed and stride length [34]. However, while the parameters in the admittance controller are fixed, they are unable to cope with the changing effects of human fatigue over time [11]. This essentially underlines the need to develop advanced optimization layers that can cope with the real-time adaptation of the control strategies [11, 20].

A representative example of the application of an admittance controller for a bilateral hip exoskeleton is provided in the work by Zhang *et al.*, where an adaptive admittance controller was implemented for an exoskeleton featuring SEA joints for both the sagittal and frontal planes [35]. The controller utilized feedback regarding interaction torques, measured through the spring deflection of the SEA, to adapt the parameters of the virtual admittance in real time, such that the exoskeleton was able to respond to balance perturbations without imposing a prespecified trajectory on the end effector [35]. Although this approach was found to be successful for balance assistance, the need for high-fidelity torque sensing for each joint is a disadvantage, as this increases the complexity and cost of the system. In contrast, the use of a velocity-proportional admittance controller, such as that presented herein, eliminates the need for torque sensing while requiring appropriate parameter tuning to ensure that the system does not either assist or impede the user at different walking speeds. The use of torque sensing, as opposed to kinematic sensing, is a branching point for the design, as the former offers a more direct measurement of the interaction force while the latter offers a lower system cost at the expense of kinematic estimation of the intent of the user.

2.5 Human-in-the-Loop Optimization

With the realization of the limitations of the fixed parameter compliant controllers, the discipline of wearable robotics was developed with the advent of HIL optimization [11, 20]. HIL optimization is a paradigm shift in the control approach from pre-programmed kinematics to real-time

physiological adaptation. In the traditional HIL approach, the human-exoskeleton system is treated as a monolithic coupled system, in which the exoskeleton assistance parameters are systematically varied while simultaneously measuring the physiological objective function, which is most commonly the metabolic cost of transport, measured via indirect calorimetry, or muscle activation, measured via sEMG [11, 20].

By utilizing an optimization method like gradient descent or covariance matrix adaptation evolution strategies (CMA-ES), HIL optimization can iteratively explore the landscape of parameters to identify the exact profile of torques required to minimize user energy expenditure precisely [20]. Landmark experiments using tethered ankle and hip exoskeletons have validated the efficacy of HIL optimization in reducing the metabolic cost of walking significantly more than general assistance [14, 20]. Nevertheless, traditional HIL optimization is plagued by an intrinsic weakness for practical application: it is extremely slow. It can take a user over 30 to 60 minutes of treadmill walking and breathing inside a metabolic mask to optimize a single parameter [14, 20]. The slow rate of convergence of traditional HIL optimization makes it impossible for it to cope with changing environmental conditions or sudden fatigue during overground long-distance walking.

2.6 Applications of Machine Learning and Reinforcement Learning

To bridge the gap between the high efficiency of HIL optimization and the need for fast and real-time adaptation, recent literature has increasingly relied on ML techniques, particularly RL and Bayesian Optimization [11, 16]. Unlike traditional HIL optimization techniques that blindly search the parameter space, an RL agent learns to develop an optimal policy to map states (for example, the user's kinematic information and gait phase) to actions (for example, specific torque values) to maximize the cumulative reward [16, 17].

Recent advances have shown the significant potential of ML-based controllers in wearable robotics. Luo *et al.* have shown that the RL-based controller, when trained using simulated scenarios and deployed on the physical hip exoskeleton, is capable of delivering metabolic savings of up to 24.3% during walking without requiring any human experiments to be performed [36]. Moreover, deep learning techniques and neural networks have been used to better estimate the gait phase and predict the user's intent using high-dimensional and noisy IMU and encoder sensor information faster and more accurately than traditional filters [16, 18].

Despite these breakthroughs, an important gap exists in the literature. Current ML-optimized exoskeletons have only been tested in highly controlled, steady-state treadmill conditions [19]. There is an urgent need to develop decentralized, hybrid control systems, where a safe, physical

control hierarchy is guaranteed by a FSM, while an RL agent is used to personalize the control signal and combat fatigue over long-distance walking [7, 19]. The motivation and focus of this thesis are to close this gap.

2.7 Research Gap

The present thesis directly addresses this gap through the development and validation of a hierarchical architecture that utilizes a FSM for deterministic and chatter-free gait phase detection as a stable sensory substrate, upon which a future Human-in-the-loop Reinforcement Learning (HIL-RL) agent can be deployed to continuously adapt the torque delivery to the shifting fatigue signature of the individual user. The present architecture therefore establishes the foundational control infrastructure required to address a problem that neither HIL nor ML approaches have yet solved in isolation.

HIL optimization approaches have provided a theoretical and experimental basis for physiological personalization of exoskeleton control. Through a coupled dynamical system approach and iterative optimization of a metabolic objective function, HIL optimization approaches have yielded the best validated reductions in metabolic cost of transport among all approaches to date [20]. However, this approach also suffers a fundamental flaw that is irreconcilable with practical deployment: convergence time. While traditional approaches to HIL optimization required a fixed 50-minute treadmill session, the more recent and widely used CMA-ES approach demands 83 ± 14 minutes of continuous walking with concurrent measurement of respiratory metabolic rate [20]. Even the Bayesian optimization approach, which converges within a fraction of this time, namely, 21 minutes, still demands that the test subject walk on a treadmill at a constant speed while also breathing through indirect calorimetry equipment [14]. This is, of course, entirely impractical outside a highly equipped laboratory setting. Moreover, after a HIL optimization approach completes, the resulting torque profiles are fixed and invariant. They are unable to adapt to the continuously increasing levels of fatigue that are a natural consequence of walking over distance. An individual whose gait kinematics change over the course of several kilometers does not benefit at all from a torque profile that was optimized for his or her fresh, unfatigued biomechanics at the start of a treadmill session [11].

The RL approach seeks to mitigate the adaptability limitation associated with static HIL control profiles through its ability to perform online, dynamic policy refinement without requiring explicit metabolic measurement [16, 17]. RL agents, in theory, are able to dynamically modify torque profiles in response to online kinematic feedback. However, in practice, recent studies confirm RL agents converge much more rapidly than classical HIL approaches. Zhang *et al.* recently presented

a study in which a least-squares policy iteration (LSPI) RL controller successfully personalized hip exoskeleton control parameters to restore a desired gait symmetry, reducing an artificially induced asymmetry index from 8.8% to -0.5% within 2 to 4 minutes of online optimization [7]. However, a subsequent study from the same research team also demonstrates a hierarchical FSM impedance and RL control approach can be used to implement task-independent personalized gait assistance under a variety of walking conditions, both in unimpaired and one post-stroke participant [19]. The validation domain for these studies is, however, highly limited, as all experiments were performed on an instrumented treadmill at a fixed walking speed, and no experiments were performed on an overground walking surface or at a range of walking speeds or for extended durations [7, 19]. The performance of RL-learned policies for the inertial changes and stride variations that are inevitable for walking over multiple kilometers is entirely unknown. Furthermore, a high-level RL agent is only able to learn if a low-level sensor system is able to detect gait phases reliably; without this, a high-level agent is forced to learn on corrupted state observations and is unsafe [18].

The gap that this synthesis has identified is therefore not a deficiency in the capabilities of either HIL or ML individually, but rather a lack of a hybrid architecture that unifies the real-time safety assurance of a deterministic low-level controller with the adaptive capabilities of a high-level learning agent, and that is specifically designed for use on an embedded platform for overground, long-distance walking applications [4, 11]. The present thesis directly addresses this gap through the proposal of a hierarchical architecture that utilizes a FSM for deterministic and chatter-free gait phase detection as a stable sensory substrate, and a Human-in-the-loop – Reinforcement Learning (HIL-RL) agent to continuously adapt the torque delivery to the shifting fatigue signature of the individual user. The present architecture is therefore a practical and specific response to a problem that neither HIL nor ML approaches have yet shown any promise for addressing.

Chapter 3 – Methodology

3.1 Hardware Design

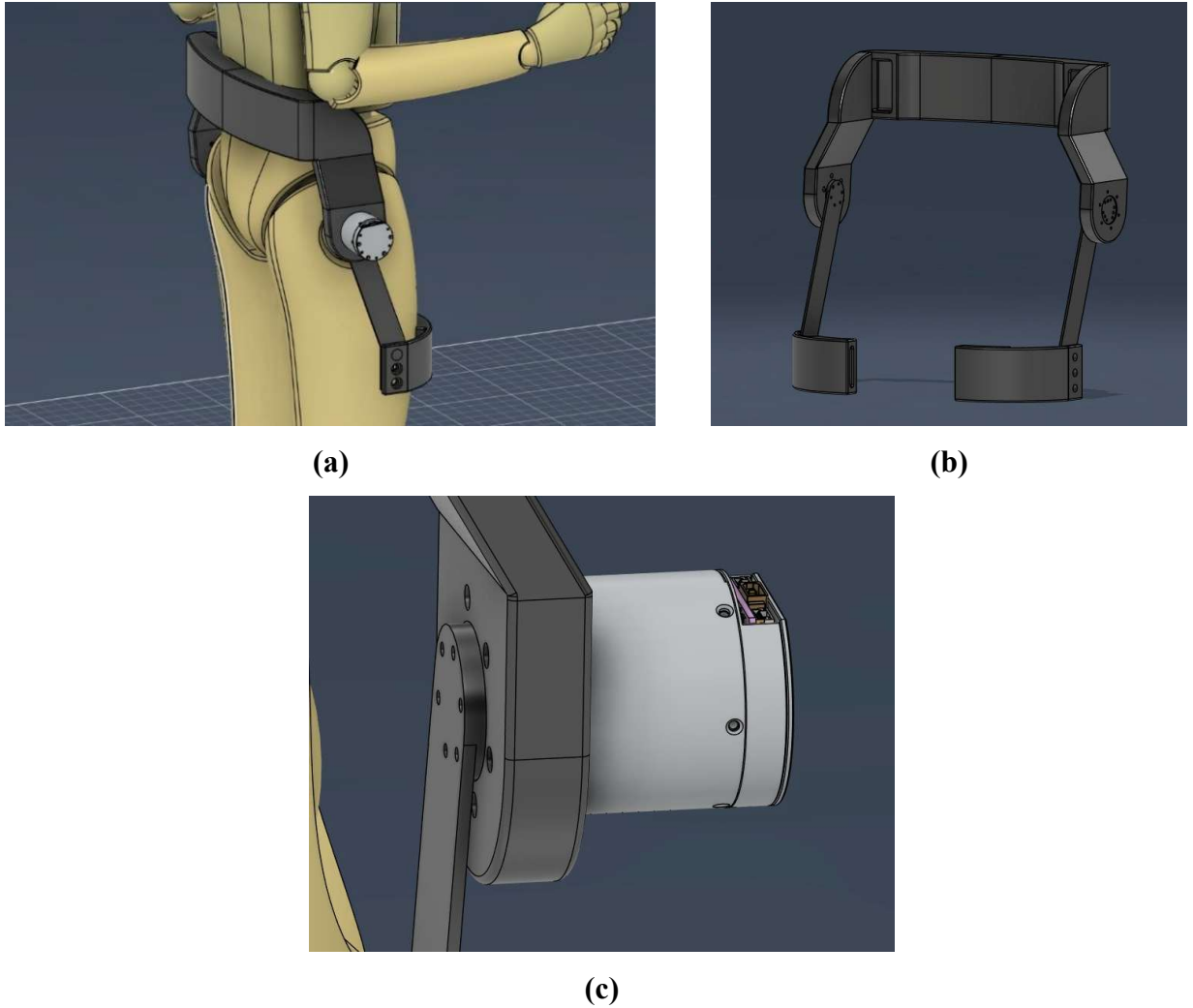


Figure 3.1. CAD model of the bilateral wearable hip exoskeleton prototype, designed in Autodesk Fusion 360. (a) Worn configuration showing the pelvic waist-binding belt, lateral rigid upright, actuation module at the hip joint, thigh linkage, and thigh cuff as positioned on the user. (b) Isometric view of the complete bilateral mechanical assembly, illustrating the symmetric dual-actuator layout and the pelvic frame structure. (c) Detail view of the actuation module showing the double-groove cable wheel mounted to the output shaft of the CubeMars AK45-10 motor.

The mechanical design of the wearable hip exoskeleton was influenced by three main mechanical objectives, which included minimization of inertial mass, accurate biomechanical alignment, and efficient mechanical transmission of torques between the actuators and the thigh [4, 8]. These objectives relate directly to the clinical application of the wearable hip exoskeleton. The inertial

mass added to the human body would cause a metabolic penalty for the user, considering that the legs would have to work to accelerate and decelerate this inertial mass with every step [4, 8]. Inaccurate biomechanical alignment would cause undesirable forces at the human-robot interface, leading to increased discomfort for the user and decreased mechanical transmission efficiency [27, 28]. Inefficient mechanical transmission would cause a wastage of the actuator's mechanical energy before it can be transferred to the user's thigh [28]. The complete mechanical assembly was designed using Autodesk Fusion 360 before it was manufactured.

3.1.1 Frame Materials and Manufacturing

To develop a highly lightweight yet adequately structural framework, the physical prototype was fabricated using 3D printing technology with PLA and ABS thermoplastics. These two materials have been selected for this particular wearable robotic device due to the combination of adequate mechanical properties for different functional areas of the wearable robotic device. PLA was selected for structural components such as larger panels and the pelvic belt, where dimensional stiffness and surface smoothness are of utmost importance. In contrast, ABS was selected for components subjected to greater mechanical stress and cyclic flexibility, such as thigh cuff interfaces and cable anchors, where its greater impact resistance and reduced brittleness reduce the risk of fractures under cyclic loading conditions [4]. These two materials have been widely accepted for wearable robotics prototyping due to their low cost, availability, and compatibility with Fused Deposition Modeling (FDM)-based rapid prototyping techniques [4]. This particular approach allows for easy changes in component geometry between different iterations of the wearable robotic device without incurring additional tooling costs, thus aligning with this incremental approach to wearable robotic device development.

The structural components of this wearable robotic device have been fabricated at a rectilinear infill density of 40% with a perimeter shell thickness of 1.2 mm, providing adequate compressive and bending stiffness for joint reaction forces while keeping mass significantly below its aluminum counterparts. The assembled components of this wearable robotic device have been bonded and fastened using M3 stainless steel hardware for its corrosion resistance and high clamping torque under multiple assembly and disassembly cycles.

3.1.2 Design Requirements

Before undertaking detailed design, a set of quantified hardware requirements was established to constrain the design space. These hardware requirements, summarized in Table 3.1, were based on normative human gait biomechanics and the performance requirements described in Chapter 1. During level ground walking at a self-selected speed, the hip joint experiences a total range of

motion of approximately 40 degrees of flexion/extension, with peak net joint moments reaching up to 80 Nm and joint velocities of up to 2.5 rad/s during the swing phase of gait [27]. The range of motion requirement of ± 30 degrees was set as a conservative minimum requirement to avoid constraining the user's natural range of motion. The requirement for an assistive torque of 5.0 Nm per joint was set to provide meaningful assistance during the swing phase without overwhelming the user's voluntary hip flexor activity, consistent with the partial assistance paradigm followed by comparable unimpaired population devices reported in the literature [27, 28]. Providing normative joint moments would require torques an order of magnitude greater, would require heavier actuators, and would likely suppress the user's voluntary neuromuscular activity, an undesirable outcome for long-distance walking where muscle activity is beneficial for metabolic efficiency [4, 8].

The total mass of the devices was set to the minimum achievable with the chosen devices, as even mass added proximally at the hip will have some quantifiable metabolic cost when locomotion is performed, even though the cost is less for each kilogram at the hip compared to the shank or foot [4, 8]. The requirement of the 500 Hz low level control rate was based on the polling rate of the Universal Serial Bus used in the ODrive Python Application Programming Interface (API). This will ensure that the updates to the motor torque command are sent at a rate much higher than the bandwidth of human gait dynamics, where most of the signal energy is found at frequencies below 5 Hz [1]. The hardware safety requirements were based on the need to provide peak current limiting, angular velocity limiting, and real-time thermal monitoring on the low-level motor controller, forming the electrical safety layer, as described in more detail in Section 3.2.

Table 3.1. Hardware design requirements for the wearable hip exoskeleton prototype.

Parameter	Requirement	Basis
Hip flexion/extension Range of Motion (ROM)	± 30 degrees minimum	Normal gait kinematics [27]
Peak assistive torque (per joint)	5.0 Nm	Simulation design target
Total device mass	Minimize (target < 3 kg)	Metabolic cost minimization [4, 8]
Actuation type	Bilateral, sagittal plane only	Gait phase assistance strategy
Low-level control rate	500 Hz	ODrive USB API sampling rate

Motor current limiting	Configurable, hardware-enforced	Safety requirement [37]
Velocity limiting	Configurable, hardware-enforced	Safety requirement [37]
Thermal protection	Automatic foldback	Safety requirement [37]

3.1.3 Mechanical Architecture

The exoskeleton is designed to be a bilateral system, giving simultaneous and symmetric assistive torques to both hip joints during the entire swing phase of walking. As shown in Figure 3.1(b), the entire mechanical system is composed of three major components: the pelvic waist binding belt, the lateral actuation system, and the thigh linkage system with cuffs.

The pelvic belt is the structural core of the system, and its main purpose is to effectively anchor the exoskeleton to the pelvis and the waist region of the user. The system is designed to distribute the resulting symmetric reaction force generated by the actuators across a large surface area, ensuring that no concentration of stress is experienced at any one point [27]. The system is also designed to counteract the resulting torque generated by the motors, which act on the thigh in the sagittal plane and is then reacted through the cable system to the pelvic belt, ensuring that no rotation or movement is experienced during walking. The belt is constructed from 3D-printed PLA panels connected with an adjustable nylon webbing strap, allowing for different waist circumferences to be accommodated. Padding is also attached to the inside surface at the areas that make contact with the iliac crest for comfort.

Extending laterally from either side of the waist belt are the two uprights, and they are made from ABS material and have a hollowed-out rectangular section to maximize the bending stiffness-weight ratio. The uprights are also adjustable in length through a slot and bolt mechanism to accommodate different lengths of users' legs. An important biomechanical function of these components is to match the mechanical rotation axis of the actuation module with the anatomical hip flexion-extension axis in the sagittal plane, as depicted in Figure 3.1(a) and (b) [27]. Misalignment between the device's kinematic axis and the biological joint axis would generate perpendicular constraint forces at the thigh interface during hip rotation, which produce joint discomfort and reduce the effective torque delivery to the thigh [27, 28].

3.1.4 Actuation System and Motor Selection

At the core of each of these actuation modules is the CubeMars AK45-10 QDD motor, whose detailed structure is presented in Figure 3.1(c). The motor is a highly integrated BLDC motor that

is coupled with a 10:1 ratio planetary gearset and a Hall-effect magnetic encoder, and is housed within a pancake-style package measuring 53 mm diameter and 43 mm axial length, weighing a total mass of 260 g for each unit [38]. The motor is rated to deliver a continuous output torque of 2.5 Nm and a maximum torque output of 7 Nm, at a rated output shaft speed of 150 rpm, giving a torque density of approximately 26.9 Nm/kg [38]. The relevant specifications for this motor are presented in Table 3.2.

Table 3.2. Key specifications of the CubeMars AK45-10 quasi-direct drive actuator [38].

Parameter	Value
Housing diameter	53 mm
Housing length	43 mm
Total mass	260 g
Gear reduction ratio	10:1
Rated output torque	2.5 Nm
Peak output torque	7 Nm
Rated output shaft speed	150 rpm
Torque density	>26.9 Nm/kg
Encoder type	14-bit magnetic
Internal driver	Present (bypassed in this design)

Figure 3.2 shows the assembled hardware prototype of the bilateral wearable hip exoskeleton in its current state. The prototype incorporates the 3D-printed PLA pelvic frame, bilateral lateral uprights, thigh cuff interfaces, CubeMars AK45-10 actuation modules, ODrive v3.6 motor controller, and associated wiring harness. The prototype demonstrates the mechanical architecture described in Section 3.1.3 and the electronics integration described in Section 3.2, and serves as the physical basis for the commissioning and closed-loop testing planned in Section 5.2.1.



Figure 3.2. Assembled hardware prototype of the bilateral wearable hip exoskeleton. The prototype comprises the 3D-printed PLA pelvic frame, bilateral lateral uprights, CubeMars AK45-10 actuation modules mounted at each hip joint, thigh cuff interfaces, and the ODrive v3.6 motor controller with associated wiring. The internal driver boards of the AK45-10 actuators are bypassed, with the three-phase stator leads and magnetic encoder connectors routed directly to the ODrive controller as described in Section 3.1.4. Full closed-loop commissioning of the assembled prototype is planned as Phase 1 of future work (Section 5.2.1).

The choice of a quasi-direct drive actuator rather than conventional high-gear-ratio motors or SEAs was motivated by the demands of admittance-controlled human-robot interaction. High-gear-ratio motors offer high motor torque, but this comes with a cost in mechanical impedance, reflected inertia, and gearbox friction, which opposes the user's natural movement during the stance phase and increases the metabolic cost of unassisted walking [34]. SEAs overcome this limitation through the use of mechanical compliance, but this comes at a cost in added weight, reduced

bandwidth, and more complicated torque control. The choice of a QDD actuator strikes a good balance between competing demands: the 10:1 reduction ratio of the AK45-10 motor minimizes reflected inertia and backdrive friction, ensuring that the joint remains transparent to the user when zero torque is required, while still allowing the 5.0 Nm required during the swing phase [34]. The transparency of the motor during the stance phase, i.e., when the motor torque demand is zero, was critical to the admittance control strategy outlined in Section 3.4, where the stance phase motor torque was explicitly set to zero, and there should be no mechanical resistance to the user's natural movement.

Although the AK45-10 comes with a built-in motor driver circuit, this internal proprietary circuitry has been disabled in this implementation. Instead, the raw three-phase electromagnetic stator leads and the output from the magnetic encoder are directly connected to an external ODrive v3.6 high-performance motor controller. The ODrive v3.6 has a maximum input voltage range from 12 to 56 V DC, a maximum peak phase current of 120 A, and FOC implemented for accurate and smooth torque control with minimal ripple current [37]. Most importantly, for safety, the ODrive also has a current limit, velocity limit, and thermal protection with foldback, and these features combined act as a safety layer in the control system implementation [37]. This allows the system to retain the desirable physical attributes associated with the AK45-10, such as its back drivability and low passive impedance, while also allowing the use of the open-source ODrive, which exposes the complete current mode torque control interface necessary for the admittance controller implementation [28, 34]. The high-level control logic, such as the FSM and admittance controller implemented in Sections 3.3 and 3.4, is implemented on a Raspberry Pi 4 single-board computer, and this communicates in real time with the ODrive via a USB interface and a 500 Hz polling rate using Python.

3.1.5 Cable-Driven Transmission

To transmit the rotational motion produced by the AK45-10 motor to the user's thigh, a custom cable-driven double-groove cable transmission was developed. Cable-driven transmissions have been extensively used in wearable robotics owing to their potential to transmit force along non-linear paths, separate the location of the actuators from the force application point, and remove the backlash problem associated with cable-based transmissions [28, 29]. However, the use of the single groove cable pulley is often accompanied by the problem of cable slippage, especially under high tension and high rate of direction change, as is the case in the heel strike to swing phase transition.

The custom-designed double groove cable pulley consists of a circular aluminum cable pulley fixed onto the output shaft of the motor, with two cable groove paths of the same radius. The high-strength Dyneema cable is wound once around each groove, and the two cable ends are fixed onto the thigh linkage. The tension in the cable is maintained at all times in the system, as the position of the hip joint is within ± 30 degrees, ensuring zero cable slip. The cable tension is split between the two cables, reducing the contact stress on each groove, thus increasing the cable durability and the potential for the system to perform over long walking durations. The cables are fixed onto the pulley and extend along the rigid upright and fixed onto the thigh linkage, thus translating the rotational motion into a direct force along the sagittal plane. The cable-driven transmission minimizes friction, does not add significant weight beyond the hip joint, and maintains the backdrivability of the QDD actuator by avoiding the use of self-locking elements in the system [12, 28].

3.1.6 Thigh Linkage and User Interface

The mechanical output of the cable transmission is connected to the user's thigh using lightweight aluminum rod linkages terminating in 3D-printed ABS thigh cuffs. Aluminum rod was used for the linkages rather than thermoplastics due to the superior specific stiffness properties, ensuring that the deflection of the linkage due to the maximum expected cable tension is less than 1 mm. The thigh cuffs are designed to fit around the anterior and lateral surface of the user's thigh, with the cable attachment point placed proximal to the knee to maximize the lever arm for hip flexion assistance. The inside surface of each cuff is lined with foam padding to ensure even pressure distribution over a large surface area to prevent pressure-induced discomfort to the user during prolonged walking and to ensure that the garment does not abrade the skin, even when worn over a standard garment [27, 30]. The cuff is secured using a single adjustable hook and loop strap.

3.2 Control Architecture Overview

For the safe and efficient handling of the complex demands required for real-time gait assistance, a distributed hierarchical structure is implemented for the control system. In a hierarchy-based wearable robot, the sensor acquisition and low-level motor command execution are separated from the high-level command execution, which is computationally intensive [39]. The execution of a wearable robot over a distance demands two different capabilities: one for high computational power to run machine learning, and the other for strict and real-time electrical reliability to ensure the safety of the user. Centralizing these two different processes on a single processor may cause timing conflicts and system instability [40]. Thus, the control strategy is divided into a high-level cognitive layer and a low-level execution layer.

Figure 3.3 presents the complete distributed control architecture. The architecture comprises three interacting tiers: Tier 1 is the high-level cognitive layer executed on the Raspberry Pi 4, Tier 2 is the low-level execution layer implemented on the ODrive v3.6 motor controller, and Tier 3 is the physical layer comprising the bilateral CubeMars AK45-10 actuators, the double-groove cable-driven transmission, and the user's hip joint.

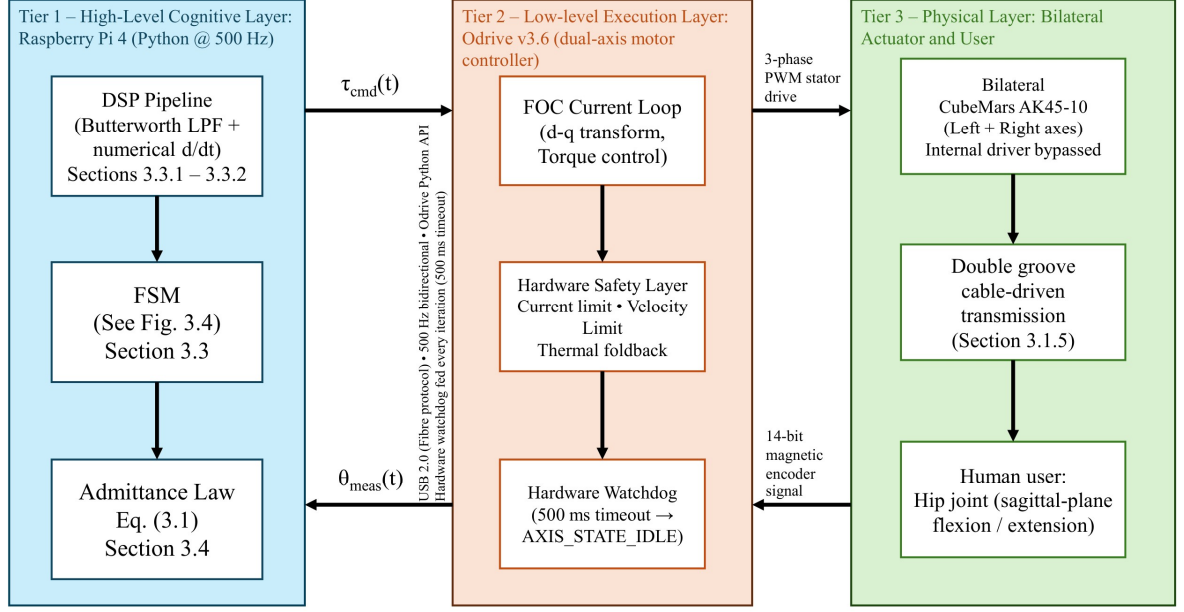


Figure 3.3. Three-tier hierarchical control architecture of the wearable hip exoskeleton. Tier 1 is the high-level cognitive layer running on the Raspberry Pi 4 at 500 Hz in Python: raw encoder samples flow through the DSP pipeline (4th-order zero-phase Butterworth low-pass filter and numerical differentiation; Sections 3.3.1–3.3.2), into the two-state FSM (Figure 3.4), and into the velocity-proportional admittance law to produce the torque command $\tau_{cmd}(t)$. Tier 2 is the low-level execution layer on the ODrive v3.6, which performs field-oriented current-loop control and enforces hardware safety limits including current limiting, velocity limiting, and thermal foldback. Tier 3 is the physical plant comprising the bilateral CubeMars AK45-10 actuators with internal drivers bypassed, the double-groove cable-driven transmission (Section 3.1.5), and the user's hip joint. The Tier 1 and Tier 2 interface operates via USB 2.0 using the ODrive Python API at 500 Hz bidirectional communication, with a hardware watchdog fed every iteration that commands `AXIS_STATE_IDLE` on a 500 ms timeout. The 14-bit magnetic encoder signal $\theta_{meas}(t)$ is returned from the motor to the Raspberry Pi via the ODrive, closing the feedback loop for gait phase detection.

The high-level cognitive layer is controlled by a Raspberry Pi 4 computer system. The Raspberry Pi 4 computer system is used as a central computing system for wearable robot systems to perform real-time sensor reading and control algorithm execution in Python [23]. The Raspberry Pi 4 is used as a hub for all Python-based control algorithms. The high-level cognitive layer is responsible for executing the FSM to determine the gait phase of the user and, in later phases, executing the RL optimizer. The FSM-based phase detection categorizes the gait cycle into distinct states, namely, the stance and the swing, using the kinematic threshold values determined from the joint angles and the angular velocities [41]. Due to the substantial computational overhead required by the RL algorithm to adapt the torque parameters according to the fatigue levels of the user [20], the Raspberry Pi is kept completely isolated from the electrical commutation of the motor.

The low-level execution is handled by an ODrive high-performance motor controller. The bypassing of the internal driver circuitry of the AK45-10 actuator, as determined in the hardware design, enables the direct interface between the ODrive and the three-phase electromagnetic coils and the high-resolution magnetic encoder of the actuator. The high-frequency control loops are handled by the ODrive, and the FOC method is adopted, wherein the feedback from the encoder is utilized to maintain the orthogonal position between the stator and the rotor magnetic fields, thereby permitting precise control over the torque while maximizing the efficiency [42]. The current, speed, and temperature foldback protection are also enabled by the ODrive, ensuring that even in the occurrence of an anomaly in the high-level Python code, the hardware is not subjected to excessive currents and temperature, thereby ensuring the stability and security of the physical actuation [37].

The vital connection between the Raspberry Pi and ODrive is achieved through a direct USB interface using the native Python API provided by the ODrive. During operation, the Raspberry Pi sends a request to the ODrive to obtain the kinematic state of the user. The ODrive responds with raw encoder values, allowing for a real-time measurement of the hip joint angle and angular velocity. This information is then used by the FSM to determine whether the user is in a stance or a swing phase. Once the kinematic state has been estimated, the Raspberry Pi computes the assistive torque command and sends it to the ODrive, which performs the actual output.

The above setup allows for a smooth implementation of the complex Python-based HIL optimization on the Raspberry Pi, along with a safe and accurate electrical actuation provided by the ODrive, ensuring a reliable system for long-distance mobility assistance.

The Raspberry Pi communicates with the ODrive v3.6 through a direct USB connection and utilizes the native ODrive Python API, which utilizes a full-speed USB 2.0 virtual serial interface

with the Fibre binary protocol. The high-level control loop is run at a sampling rate of 500 Hz, resulting in a nominal inter-sample period of 2 ms. The 500 Hz sampling rate is dictated by the maximum polling rate of the ODrive USB API and is consistent with the real-time communication rates used in comparable wearable exoskeleton systems. This rate satisfies the Nyquist criterion for the 5 Hz filter cutoff by a factor of fifty, providing substantial margin against aliasing of the encoder velocity signal. The sampling rate is consistent with the real-time communication rates used for other wearable exoskeleton and robotic teleoperation systems that require synchronization of kinematic feedback [39, 43]. The 500 Hz sampling rate means that each sampling period, the ODrive is polled for the current position and velocity estimates, the state transition logic and torque calculation are performed, and the resulting torque command is sent to the ODrive through the API's `input_torque` attribute. The end-to-end computation time for this process, including the time to perform USB transactions and execute Python code, is limited to the 2 ms inter-sample period. The end-to-end actuation time for a wearable exoskeleton system is required to be significantly less than 100 ms to maintain natural responsiveness and prevent destabilization of the system [44]; the 2 ms sampling period is well within this range.

The configurable hardware-level watchdog timer is provided at the firmware level for the ODrive to mitigate the risk of communication disruption between the Raspberry Pi and the ODrive [37]. The watchdog timer is configured to be enabled for the ODrive through the `axis.config.enable_watchdog` parameter, and a configurable timeout is provided for this watchdog, configured to 500 ms for this system. The high-level Python control loop continuously feeds the watchdog timer every iteration through the `axis.watchdog_feed()` command immediately after sending the torque command. If the USB connection is lost, if the Raspberry Pi process crashes, or if the high-level Python loop is stuck for any reason, the watchdog timeout is exceeded, and the ODrive firmware autonomously drives both axes to `AXIS_STATE_IDLE`, sending a command for zero motor current to the motors [37]. This ensures that a software failure in the high-level Python layer does not allow an uncontrolled torque output to the hip joint. The 500 ms timeout is sufficiently high to allow for transient and brief USB connection spikes while still ensuring that a prolonged unintended torque output is avoided. In the event that the ODrive is disarmed due to a watchdog timeout, an error flag is latched that must be cleared through the `axis.clear_errors()` command before a new closed-loop control is initiated.

Before entering closed-loop torque control, each axis needs to go through a mandatory startup calibration routine, which is implemented within each axis's state machine [37]. The default state after a power-up is `AXIS_STATE_IDLE`, where the motor is off, and the encoder is uninitialized. The startup calibration routine follows this order: The motor resistance and inductance calibration

routine is initiated by setting `axis.requested_state = AXIS_STATE_MOTOR_CALIBRATION`. In this routine, a series of voltage pulses, known as test voltages, are applied to each phase of the motor, and the current response is measured, allowing the resistance and inductance values per motor phase to be determined. These values are required by the current controller, which implements FOC, in order to linearize the motor's electrical model. The second step in the calibration routine is the encoder offset calibration routine, where `axis.requested_state = AXIS_STATE_ENCODER_OFFSET_CALIBRATION`. In this routine, the motor stator is energized, and a constant electrical angle is maintained, with the rotor locked in place magnetically, after which the motor shaft is turned through a specified mechanical angle, allowing the electrical phase reference frame's angular offset relative to the mechanical zero position, defined by the incremental encoder's zero index, to be determined [37]. The AK45-10 motor uses a 14-bit absolute magnetic encoder, which provides 16,384 unique position counts per revolution. However, the ODrive firmware requires the encoder offset calibration routine to be repeated at every power-up, because the angular offset between the encoder mechanical zero and the motor electrical phase reference frame is a property of the motor-encoder assembly and cannot be recovered from the encoder output alone after a power cycle. To avoid having to perform motor resistance calibration every time the motor is powered up, the results obtained during the first calibration routine are stored in flash memory using `axis.motor.config.pre_calibrated = True` and `odrv.save_configuration()`. The only calibration routine that needs to be repeated after every startup is the encoder offset calibration routine. After both calibration routines have been completed, the axis state machine will switch to `AXIS_STATE_CLOSED_LOOP_CONTROL`, where the control mode will be initialized to `CONTROL_MODE_TORQUE_CONTROL`, and the Raspberry Pi will be able to issue torque setpoints, which will be passed to the axis's controller object's `axis.controller.input_torque` attribute.

3.3 Gait Phase Detection (FSM)

The essential pre-requisite for transparent and effective robotic assistance is the detection of the current gait phase of the user. If assistive torque is applied at an incorrect time, it will disrupt the natural biomechanics of the user, increasing the metabolic cost and compromising balance [15]. To enable precise synchronization without the need for cumbersome external wearable sensors, the proposed system will derive the gait phase entirely from the AK45-10 actuator's internal high-resolution magnetic encoder.

Using the native Python API, the Raspberry Pi will continuously poll the ODrive for the absolute angular position of the motor shaft. Since the actuator is rigidly aligned with the natural human hip joint in the sagittal plane, the encoder signal will directly represent the user's hip flexion and

extension angle. However, the raw position signal does not provide information on the direction of movement. Moreover, calculating the discrete time derivative of the signal to obtain the angular velocity will inherently amplify high-frequency sensor noise. Therefore, a DSP pipeline is also implemented in Python. Here, a fourth-order zero-phase Butterworth LPF with a cutoff at 5 Hz is used to isolate the natural frequencies of human locomotion while filtering out unwanted noise.

The filtered angular velocity is used as the primary control variable for a two-state FSM. For the baseline control architecture, the FSM will reduce the human gait cycle into two basic modes: the Stance Phase (State 0) and the Swing Phase (State 1). Biomechanically, the beginning of the swing phase, where the leg is lifted and swung forward, is defined by hip flexion, which is represented by positive angular velocities. Conversely, during the stance phase, the foot is planted on the ground with the torso translating forward over the leg, defined by hip extension and negative angular velocities [9].

Figure 3.4 presents the block diagram of the complete gait-phase-detection pipeline, from the raw AK45-10 encoder signal through the DSP chain to the two-state FSM and the resulting assistive torque command. The dashed boundary in the diagram encloses the FSM itself, whose state-transition logic is formalised in the remainder of this section.

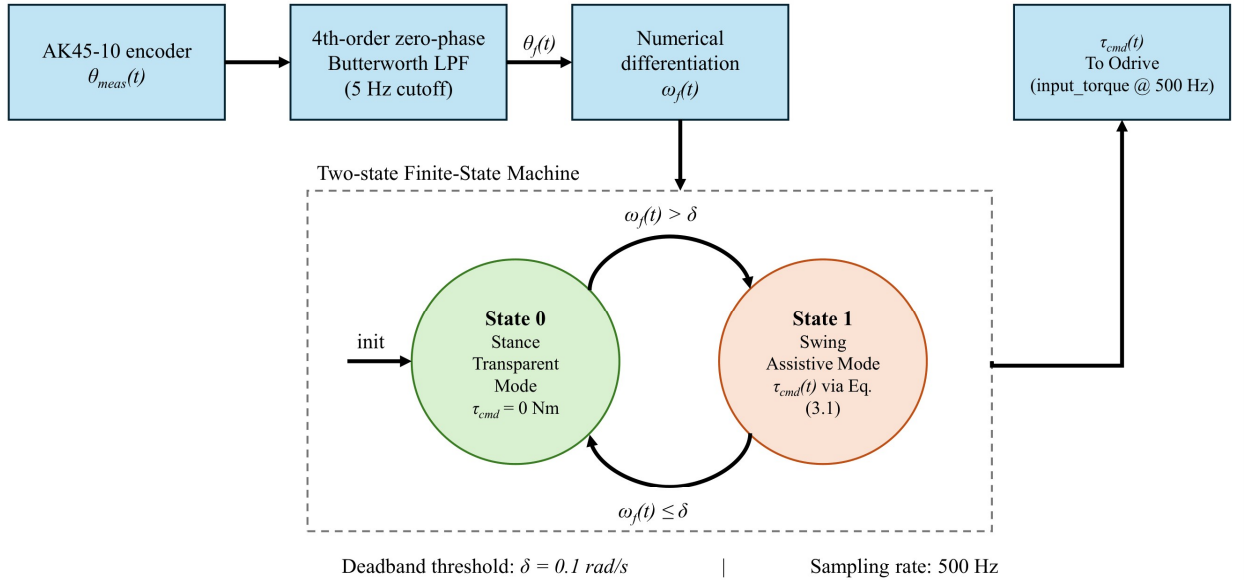


Figure 3.4. Block diagram of the gait-phase-detection pipeline and its two-state FSM. The AK45-10 internal magnetic encoder signal $\theta_{meas}(t)$ is passed through a 4th-order zero-phase Butterworth low-pass filter with a 5 Hz cutoff (Section 3.3.2), yielding the filtered position signal $\theta_f(t)$, which is then numerically differentiated to produce the filtered angular velocity $\omega_f(t)$. The

FSM (dashed enclosure) uses $\omega_f(t)$ to toggle between State 0 (Stance, Transparent Mode, $\tau_{cmd} = 0$ Nm) and State 1 (Swing, Assistive Mode, τ_{cmd} computed via the admittance law). Transitions are governed by a single velocity deadband threshold $\delta = 0.1$ rad/s (Section 3.3.3): the system enters State 1 when $\omega_f(t) > \delta$ and returns to State 0 when $\omega_f(t) \leq \delta$. The resulting torque command $\tau_{cmd}(t)$ is transmitted to the ODrive at 500 Hz via the ODrive Python API (Section 3.2).

In order to manage the transitions between these two states, strict velocity-based conditional logic, along with a safety deadband, is employed. The state transition logic is defined as follows:

- Transition to State 1 (Swing Phase): If the filtered angular velocity exceeds a positive threshold ($\omega > 0.1$ rad/s), then the FSM concludes that the user has initiated a forward step.
- Transition to State 0 (Stance Phase): If the angular velocity falls below the threshold ($\omega \leq 0.1$ rad/s), indicating the leg is either moving backward during ground contact or that the user is standing completely still.

The use of this 0.1 rad/s deadband threshold is a critical component in the design of the state machine, which is commonly referred to as "control chattering." The use of this powerful, lightweight state machine approach ensures that real-time kinematic synchronization between the user and the exoskeleton, thereby providing a solid foundation upon which the admittance control and Machine Learning optimization functions operate.

3.3.1 Signal Frequency Content and Filter Cutoff Rationale

The reasoning behind the 5 Hz cutoff is based on the well-documented frequency content of the human lower extremity kinematics during walking. The majority of the signal energy is well documented to reside below 6 Hz, with the high-frequency components being the result of measurement noise and soft tissue artefacts, as opposed to actual skeletal motion [45, 46]. The 5 Hz cutoff maintains the entire fundamental frequency and its significant harmonics while also maintaining a 1 Hz guard band below the commonly referenced 6 Hz reference found in published gait biomechanics data [45, 46]. The angular velocity signal is the result of the numerical differentiation of the position signal, and the noise amplitude is proportional to the signal frequency. The LPF is necessary to ensure the signal is appropriate for threshold-based state determination [47].

3.3.2 Filter Order and Zero-Phase Implementation

The 4th-order Butterworth filter was chosen for the DSP pipeline for two different, yet complementary, reasons. First, the maximally flat passband response ensures that the amplitude of low-frequency kinematic content is preserved without any ripples, a requirement for the velocity-

proportional law described in Section 3.4. Secondly, a 4th-order filter roll-off rate of 80 dB/decade is sufficient to reduce quantization noise and motor commutation components, which for the AK45-10 actuator and a 10:1 gear ratio may reach frequencies well above 10 Hz and are periodic during steady walking. The use of 4th-order Butterworth filters, configured for zero-phase response and forward and backward filtering (`scipy.signal.filtfilt`), is the established norm for kinematic signal conditioning for biomechanics research [47, 48]. The zero-phase response is obtained by filtering the signal forward and then backward, effectively doubling the filter order to 8 (twice the nominal 4) and ensuring that no time shift is introduced into the velocity calculation, which is a requirement for a threshold-based FSM because any time shift would mean that the state transition would be recorded at a time different from the actual event, resulting in systematic phase errors for torque application [48].

3.3.3 Deadband Threshold Selection and Comparative Rationale

The value of the 0.1 rad/s deadband was determined from an analysis of the noise floor in the simulated encoder signal after the application of the filter. In the simulation, the standard deviation of the residual noise in the velocity signal during the stationary interval was measured, and the threshold value was set to exceed the noise floor by a factor of approximately three, in keeping with standard practice in binary threshold classifiers. To illustrate the sensitivity of the FSM to the choice of the threshold value, the performance of the algorithm was simulated using three different values: a zero-crossing threshold value of 0.0 rad/s, the selected deadband value of 0.1 rad/s, and a conservative value of 0.3 rad/s. The zero-crossing threshold value caused repeated false state transitions during the stationary interval, directly illustrating the chattering effect. The 0.3 rad/s value eliminated the chattering effect but caused a noticeable delay in the detection of the onset of the swing phase compared to the kinematic event, which in turn caused a reduction in the percentage of the swing phase under the control of the algorithm. The 0.1 rad/s value eliminated the chattering effect while allowing prompt detection of the onset of the swing phase. A similar approach to the use of a velocity threshold in the gait phase detection algorithm in hip exoskeletons using an encoder has been reported in the recent literature, confirming the viability of the proposed approach to the use of an angular velocity deadband in the encoder signal from the 500 Hz sampling frequency [48].

3.3.4 FSM State Transition Summary

For completeness, the state transition logic of the two-state FSM can be written in formal terms. The state machine processes the filtered hip angular velocity $\omega_f(t)$ sampled at 500 Hz. On each sampling occasion, the state machine will:

- Transition from State 0 (Stance) to State 1 (Swing) and turn on the admittance control, if the current state is State 0 (Stance) and $\omega(t) > \delta$;
- Transition from State 1 (Swing) to State 0 (Stance) and set the motor torque command to zero, if the current state is State 1 (Swing) and $\omega(t) \leq \delta$;

where $\delta = 0.1$ rad/s is the deadband value. No explicit time-based state transition conditions or hysteresis are included in the state machine, and all transitions are instantaneous in the current implementation. This minimalist approach has been adopted in the interests of computational simplicity on the Raspberry Pi platform, as well as interpretability of system operation in the initial stages of hardware commissioning. The addition of state transition conditions based on the duration of stay in each state, or the incorporation of additional sensor confirmatory logic, is identified as a direction for future refinement once overground walking trials have characterized the noise environment in which the deployed system will operate.

3.4 Admittance Control Strategy

The next step after the accurate identification of the gait phase through the FSM is to determine the level and time of application of the assisting force. In wearable robotics, the use of a stiff trajectory-tracking strategy (position control) often results in the user's limb moving along a predefined path [10, 49]. This may cause why a human and the robot to act against each other, resulting in discomfort for the user and an increased level of metabolic rate [10, 49]. To avoid such a situation, an admittance control strategy is used for the proposed wearable robot. Admittance control is a virtual interface that behaves like a mechanical system and is highly compliant, calculating a torque output (τ) based on the input provided by the human (ω) [34]. The user is therefore always in full control of the step length and walking speed, while the robot assists the movement with a synchronized and assisting push [34].

The admittance controller is responsible for governing the behavior of the CubeMars AK45-10 motor based on the active state provided by the FSM, resulting in two different operational modes:

3.4.1 Transparent Mode (Stance Phase, State 0):

During this stance phase, when $\omega \leq 0.1$ rad/s, the foot remains in contact with the ground, supporting the weight of the user and facilitating forward propulsion. The introduction of artificial torque during this weight-bearing period can interfere with postural stability and hence modify the original path of the CoM. Therefore, during this period, when the FSM is in State 0, the commanded motor torque, i.e., τ_{cmd} , should be strictly 0.0 Nm. This choice of zero commanded

torque, considering the QDD nature and relatively low gear reduction ratio of the AK45-10, makes the joint actuator highly back-drivable [34]. This allows the user to experience near-zero mechanical resistance during the stance phase, thereby providing a natural and unobstructed stance phase (Transparent Mode) [34].

3.4.2 Assistive Mode (Swing Phase, State 1):

As soon as the user initiates a step and enters State 1, i.e., when $\omega > 0.1$ rad/s, the controller enters Assistive Mode, and its goal is to assist the hip flexor muscles in swinging the leg forward by injecting positive mechanical work [50]. However, to make this assistance feel natural and smooth, a step change in torque, i.e., a step function, is not used. Instead, a dynamic torque profile, directly proportional to the instantaneous filtered angular velocity, is provided by the admittance controller, simulating a virtual assistive damper [34].

The baseline commanded torque is calculated using the following linear velocity-proportional equation:

$$\tau_{cmd}(t) = \left(\frac{\omega(t)}{\omega_{max}} \right) \times \tau_{peak} \quad (3.1)$$

Where:

- $\tau_{cmd}(t)$

is the instantaneous torque command sent to the ODrive over the CAN/USB interface.

- $\omega(t)$

is the real-time filtered angular velocity of the hip joint.

- ω_{max}

is the normalized maximum expected angular velocity during a typical swing phase.

- τ_{peak}

is the maximum allowable assistive torque (e.g., 5.0 Nm), establishing a safe upper limit for actuation.

Under this baseline admittance paradigm, the exoskeleton pushes hardest in the middle of the swing phase (when the leg is swinging the fastest) and smoothly tapers off as the foot prepares for a heel strike [34].

3.4.3 Derivation and rationale of $\omega_{\max}$

The normalization parameter $\omega_{\max}$ in Equation (3.1) is not an empirically determined value, but rather an analytically derived value based on the sinusoidal model of the user's movement. The model is given by the following equation, where the hip angle is represented as $\theta(t) = A \cdot \sin(2\pi f \cdot t)$, and the parameters are set to $A = 20^\circ$ and $f = 1.0$ Hz. The angular velocity is given by $\omega(t) = A \cdot 2\pi f \cdot \cos(2\pi f \cdot t)$, and the theoretical peak value is $2\pi \cdot 20^\circ \approx 125.7$ deg/s, which converts to approximately 2.19 rad/s. The value of $\omega_{\max}$ is determined as the maximum value found over the entire trial in the filtered signal, and the value is within the range of the angular velocities found in the hip during walking on level ground at self-selected velocities in healthy adults, ranging from 60 to 160 deg/s depending on the walking speed [51]. The scaling factor ensures that the admittance torque reaches the peak value τ_{peak} only when the user's movement is within the biomechanically expected maximum extent of the arc of the swing.

3.4.4 Saturation behavior and safe operating limits

The control law defined in Equation (3.1) is linear for all values of ω in the range $[0, \omega_{\max}]$. The case where the angular velocity precisely equals $\omega_{\max}$ results in a saturation value $\tau_{\text{peak}} = 5.0$ Nm. If the user walks faster than the simulated model predicts, causing the filtered encoder velocity to temporarily surpass $\omega_{\max}$, then the ratio $\omega(t)/\omega_{\max}$ will be greater than one, causing τ_{cmd} to be greater than τ_{peak} . However, this case is addressed through hard saturation, where the ODrive has limits on current and velocity, regardless of what the Python command specifies [37]. The result is that, should the admittance equation result in a torque request greater than the specified limit, the phase current will be limited by the ODrive, effectively capping the torque at some hardware-defined ceiling. The chosen ceiling of 5.0 Nm remains below the actuator's rated peak torque of 7.0 Nm [37], avoiding sustained overload operation. Although 5.0 Nm exceeds the AK45-10 continuous rating of 2.5 Nm, the motor does not sustain this output across the full gait cycle. Assistive torque is applied exclusively during the swing phase, which constitutes 48.6% of the stride period under the sinusoidal kinematic model used in this work. The duty-cycle-averaged torque across a complete stride is therefore $\tau_{\text{avg}} = 2.95 \times 0.486 \approx 1.43$ Nm, which is 43% below the continuous rating of 2.5 Nm. This confirms that mean thermal loading remains within the motor's rated continuous operating envelope, ensuring thermally sustainable operation during long-distance walking [37].

3.4.5 Integration with Future Machine Learning Optimization:

Though this basic admittance controller will effectively provide user-injected assistive energy and guarantee user safety, a fixed value of τ_{peak} and a static form of the admittance profile are metabolically suboptimal for long-distance walking [11]. Indeed, as a user starts to exhibit signs

of fatigue, their natural ω and stride dynamics will deviate. In the next iteration of this work, a HIL-RL agent will be utilized on the Raspberry Pi. This ML component will continuously monitor deviations in user kinematics and optimize τ_{peak} and the profile of the admittance controller in real-time, adjusting the exoskeleton's assistance profile to minimize user metabolic fatigue over long distances [7, 11, 19].

3.4.6 Sensitivity of τ_{peak} and justification of the chosen value

The value of τ_{peak} essentially functions as a scaling factor for all of the admittance curve. Increasing this value will proportionally increase all assistive torques, regardless of swing velocity, while decreasing it will proportionally decrease all assistive torques. At a value of 5.0 Nm, it was determined in Section 4.2 that the amount of mechanical energy injected per stride is 2.38 J, and the mean swing-phase torques have an average magnitude of 2.95 Nm. For comparison, it was determined in [1] that the net positive work done per stride by biological hip flexors during level ground walking under no load conditions is between 0.15–0.25 J/kg body mass, or about 10–18 J for a 70 kg person. The amount of energy injected per stride by the proposed exoskeleton is thus seen to be a supplementary value, rather than a replacement for biological muscular action, in keeping with its assistive, rather than rehabilitation-oriented, design. Values of τ_{peak} below about 2.5 Nm would be considered negligible relative to biological hip flexor action and thus have low therapeutic value. Values above the 7 Nm peak torque rating of the AK45-10 would be considered unsafe due to potential for sustained thermal overload [37]. The selected value of 5.0 Nm exceeds the 2.5 Nm continuous rating but remains within the 7 Nm peak rating; the mean swing-phase torque of 2.95 Nm, applied only during the 48.6% swing fraction of each stride, produces a duty-cycle-averaged thermal load below the continuous rating, making sustained operation at 5.0 Nm peak mechanically feasible [38].

Chapter 4 – Preliminary Results and Simulation

4.1 Kinematic and Energetic Simulation Results

The simulation models a continuous human walking cycle at an approximate physiological frequency of 1.0 Hz and oscillates between 20 degrees of hip flexion and extension as a simplified sinusoidal kinematic model [8]. To emulate real-world conditions from the CubeMars AK45-10 internal encoder, random Gaussian noise was superimposed onto the true kinematic trajectory. Figure 4.1 illustrates the system's real-time response across four critical domains.

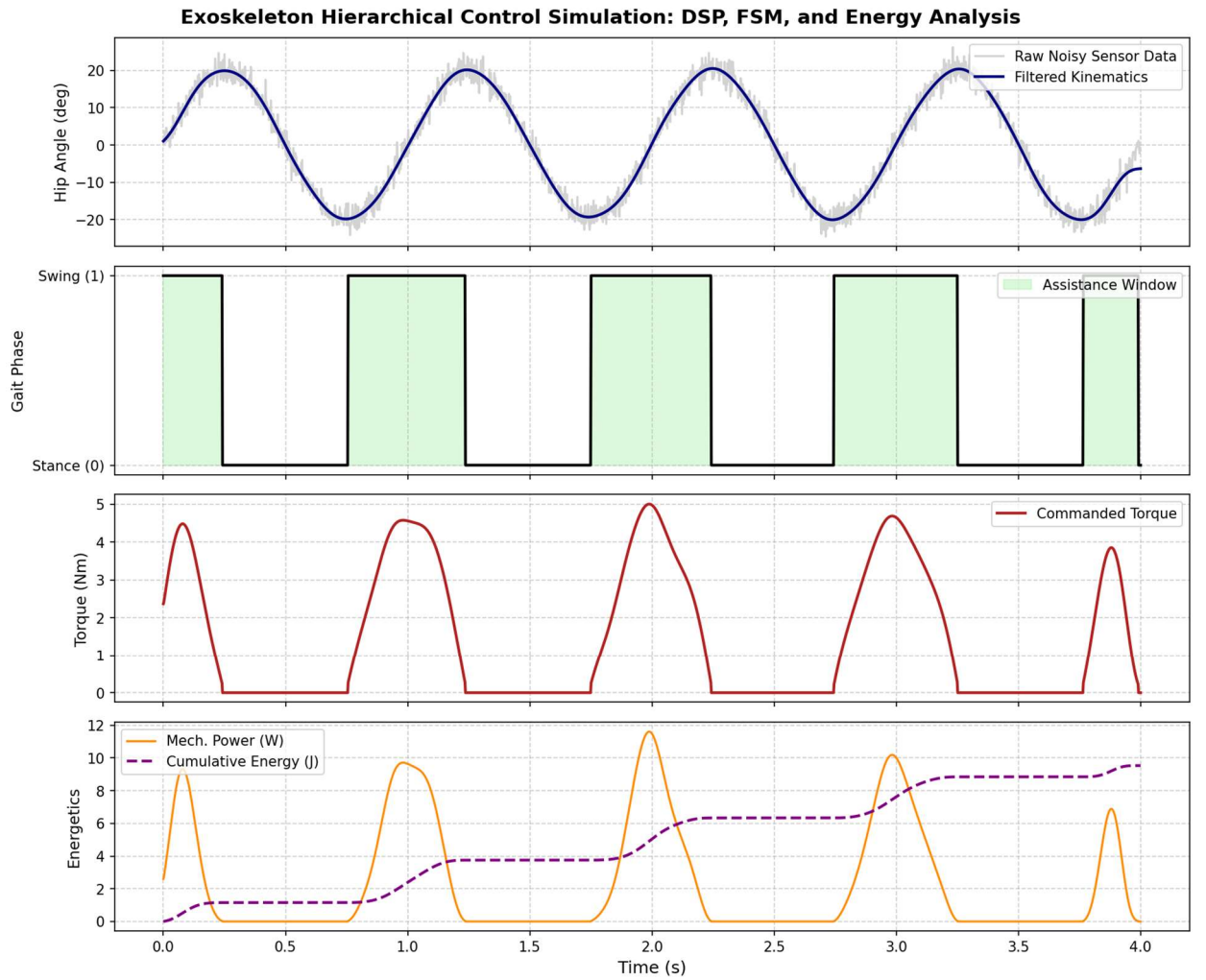


Figure 4.1: Simulation of the hierarchical control architecture. (Top) Raw noisy encoder data conditioned via a 4th-order Butterworth LPF. (Second) FSM state estimation accurately isolating the swing phase (shaded green). (Third) Velocity-proportional admittance torque command peaking at 5 Nm. (Bottom) Mechanical power and cumulative energy integration proving positive work injection into the gait cycle.

4.1.1 Signal Processing and State Estimation:

As depicted in the top panel of Figure 4.1, the raw sensor data (light gray) has high frequency noise, which would lead to instability in the traditional derivative-based velocity control algorithm. The success in the implementation of the 4th-order zero-phase Butterworth LPF filter (5 Hz cutoff) in recovering the true kinematic trajectory (navy blue) without the introduction of phase lag is clear [1, 52]. The kinematics of the human joint movement during walking are dominated by frequencies below 5 Hz, which are successfully recovered from the noisy signal, and the zero-phase forward and backward implementation of the filter ensures no group delay, which would otherwise interfere with the accuracy of the velocity signal [52].

The second panel of Figure 4.1 clearly illustrates the robustness of the proposed FSM control algorithm. The ability of the algorithm to successfully recognize the start of the hip flexion movement using the filtered signal is clear, as is the success in the implementation of the 0.1 rad/s deadband threshold in separating the swing phase (State 1, shaded green) from the stance phase (State 0).

4.1.2 Torque Generation and Biomechanical Energetics:

The third panel validates the admittance control strategy. The commanded torque is strictly 0.0 Nm during the entire stance phase, ensuring that the robot is in transparent mode and that the human is responsible for carrying the entire weight. As the system transitions into the swing phase, a smooth and velocity-proportional assistive torque is provided that peaks exactly at the desired 5.0 Nm safety limit at maximum swing velocity, and smoothly decreases before heel strike.

Most importantly, the bottom panel mathematically validates the fundamental hypothesis presented in this thesis: that of energy efficiency. The mechanical power was computed as the product of the instantaneous commanded torque and the angular velocity ($P = \tau \cdot \omega$). The integration of this power over time is the cumulative mechanical energy injected into the system (dashed purple curve). The strictly positive power curve and the increasing cumulative mechanical energy curve prove that the baseline admittance controller is effectively and consistently injecting assistive mechanical power into the human gait cycle [14].

This simulation demonstrates that the low-level architecture is fundamentally sound, highly stable, and ready for deployment onto the Raspberry Pi and ODrive hardware. The next step for this research is to use this validated baseline as the starting point for the RL optimizer, which will dynamically optimize the peak torque and admittance values to maximize the energetic savings presented in Figure 4.1 [8, 14].

Table 4.1: Summary of simulation performance metrics for the baseline admittance controller. All values are derived from the sinusoidal kinematic model at 1.0 Hz stride frequency, 5.0 Nm peak torque design input, and 0.1 rad/s FSM deadband threshold.

Metric	Value	Notes
Peak assistive torque	5.0 Nm	Design input; AK45-10 rated peak 7 Nm, continuous 2.5 Nm
Mean swing-phase torque	2.95 Nm	Averaged over all swing-active samples
Stride-averaged injected energy	2.38 J/stride	Total 9.54 J accumulated over the 4-second simulation window (three complete detected strides)
Peak mechanical power	11.6 W	Instantaneous max of $P = \tau \cdot \omega$
Root Mean Square (RMS) mechanical power	4.20 W	Root-mean-square over full simulation window
Swing phase fraction	48.6%	Fraction of stride samples in FSM State 1

The quantitative values in Table 4.1 provide a characterization of the energy delivery profile of the baseline admittance controller under the sinusoidal kinematic model. The average swing phase torque of 2.95 Nm is only 59% of the 5.0 Nm peak, which is expected due to the velocity-proportional control law, which only commands the maximum torque at the instant of maximum hip flexion velocity, while the average torque is much lower. The 4.20 W value of the RMS mechanical power, compared to the 11.6 W instantaneous maximum, reflects the duty-cycle effect of the FSM, which only acts during the 48.6% swing fraction of the gait cycle, while the exoskeleton is completely passive during the stance fraction. The average injected energy of 2.38 J/stride is within the range of the values reported in the literature for exoskeletons designed to provide metabolic benefits during walking, which can range from 0.5 J to 4.0 J/stride depending on the level of assistance and the joint being assisted [8, 14]. As mentioned in the previous paragraph, all the values in Table 4.1 are obtained from the simplified sinusoidal kinematic model, which are not based on measured human gait data but are rather the theoretical performance of the admittance controller under the specified conditions.

4.2 FSM Validation and Deadband Implementation

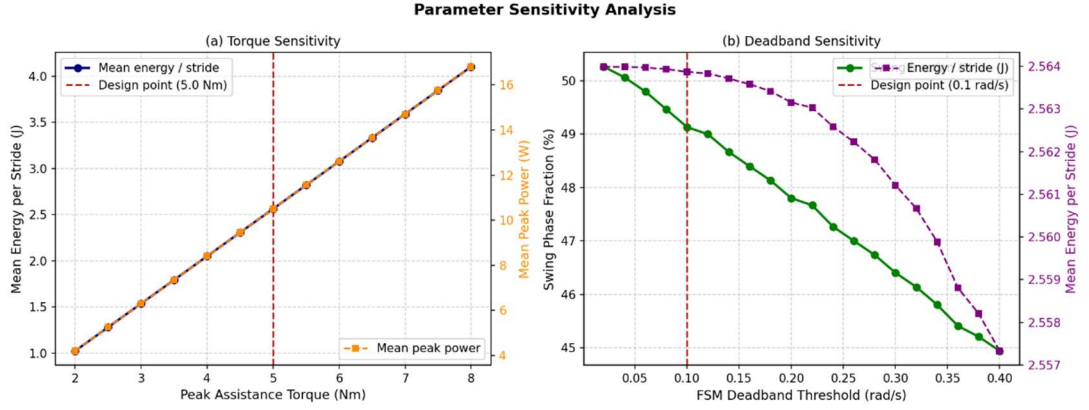


Figure 4.2: Parameter sensitivity analysis illustrating the effect of varying the FSM deadband threshold and peak assistive torque on energetic output. The sweeps characterize the admittance controller's response to parameter variation, comparing a strict zero-crossing threshold against the selected 0.1 rad/s deadband and a conservative 0.3 rad/s threshold to demonstrate the prevention of control chattering without introducing excessive phase delay.

The second panel of the simulation (Figure 4.1) verifies the robustness of the FSM in identifying the swing phase of the human gait cycle, a technique commonly adopted in the control of lower limb wearable robotics [49]. Although, in an ideal and noise-free world, the transition from the stance phase to the swing phase could be theoretically determined as the point where the angular velocity crosses zero (0 rad/s), in the real world, this naive method is extremely dangerous. In the case where the user is stationary, moving their weight, or is about to make a walking action, the internal motor encoder will detect the movements. Even after the data is filtered using a digital LPF, as the zero-lag Butterworth filter is often adopted to filter the data collected from the kinematic sensors [22], the data could still oscillate rapidly above and below zero.

If the FSM uses a hard zero-crossing criterion, such variations would cause "control chattering," where the exoskeleton would rapidly and violently switch between ON and OFF states. In mechatronics, severe control chattering would result in unwanted oscillations, instabilities, and saturation, which would result in considerable battery drain and electrical spikes [53]. To address this, a kinematic safety deadband of $\pm \varepsilon$ has been incorporated into the state machine logic. The approach follows heuristic threshold-based guidelines that are critical in state classification in real-time, avoiding erroneous state transitions due to signal noise [54]. As shown in the simulation, the state machine will strictly evaluate the velocity against this deadband. The green shaded areas in Figure 4.1 confirm that State 1 (Assistive Swing Phase) will only be activated if the user shows a clear, deliberate, and sustained movement in the forward direction, crossing the threshold.

Similarly, as the leg reaches peak flexion and prepares to strike the ground, the velocity will diminish. The state machine will pre-emptively shut off the motor assistance as the velocity dips back below $-\varepsilon$, returning to State 0 (Transparent Stance Phase) in a clean, square wave fashion. The simulation shows that there is absolutely no chattering or flickering whatsoever at the boundary conditions. This proves that the state machine, together with the deadband, has effectively filtered out unwanted movements, ensuring that motor torque will only be injected when biomechanical intent has been definitively established [49, 54].

4.3 Biomechanical Energy Analysis

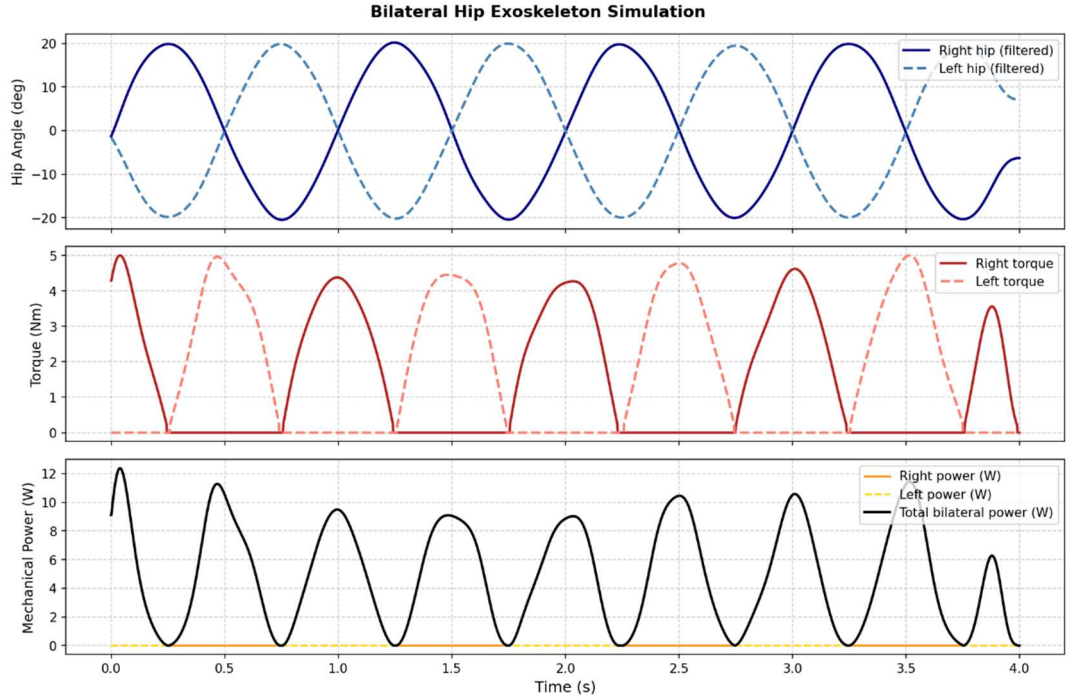


Figure 4.3: Biomechanical energy analysis of the exoskeleton-assisted gait cycle. The plot details the instantaneous mechanical power ($P = \tau_{cmd} \cdot \omega$) and the resulting cumulative mechanical energy injected over time. The strictly positive power curve confirms that the admittance controller delivers up to 2.38 J per stride, successfully offloading biological hip flexor effort without resisting the user's natural movements during the stance phase.

The ultimate goal of the present thesis is the reduction of human fatigue during long-distance walking. The energy exchange between the exoskeleton and the human must be considered to demonstrate controller effectiveness.

The bottom panel of Figure 4.1 provides a rigorous biomechanical analysis of the energy exchange between the exoskeleton and the human.

Instantaneous mechanical power (P , in Watts) supplied to the human hip joint during a rotational movement can be found by multiplying the torque command signal (τ_{cmd}) by the corresponding angular velocity (ω) of the hip joint:

$$P(t) = \tau_{cmd}(t) \cdot \omega(t) \quad (4.1) [1]$$

If the power is positive, then the exoskeleton is doing positive mechanical work, assisting the muscles. If the power is negative, then the exoskeleton is doing negative work, effectively absorbing power and therefore resisting the movement of the user. As shown in the orange curve in Figure 4.1, the power is strictly positive for the entire swing phase, reaching its maximum at the exact same time as the maximum velocity. The strict enforcement of a 0.0 Nm torque command for the entire stance phase, where the commanded torque is zero, inherently ensures that no negative power is generated. Mathematically, this ensures that the exoskeleton does not "fight" the natural biomechanics of the user, an important consideration to prevent metabolic waste [15].

To determine the overall level of assistance provided over time, the numerical integration of the power was performed to determine the cumulative mechanical energy (E , measured in Joules) supplied to the gait cycle:

$$E = \int_0^t P(t)dt \quad (4.2)$$

The dashed purple line in Figure 4.1 demonstrates this concept of cumulative energy. As each successive phase of swinging occurs, the overall energy injected increases in a step-like fashion. In a biological system, every Joule of mechanical work provided to the system through the exoskeleton actuator means one less Joule of work that the human hip flexor muscles, such as the iliopsoas and rectus femoris, do not have to perform [55]. Over the course of a long-distance walk, spanning thousands of strides, this constant injection of positive mechanical energy directly relates to a reduction in the overall human cost of transport.

Although this baseline simulation has successfully proved the efficacy of the FSM and admittance controller in injecting positive energy, the fixed parameter of peak torque has yet to be optimized for different levels of human fatigue. This RL agent, however, will use this baseline system to dynamically control the torque curve and maximize the slope of the line representing cumulative energy, while minimizing electrical power consumption from the ODrive, thereby achieving true HIL energy efficiency.

In an effort to understand how this simulated energy injection relates to previously published research, a discussion on how this value relates to values published by similar hip-assistive devices

is warranted. The biological net positive mechanical work performed by the human hip flexor muscles during unloaded level-ground walking has been estimated to be 0.15-0.25 J/kg per stride, or approximately 10-18 J for a 70 kg human adult [1]. The 2.38 J injected per stride by the simulated exoskeleton, therefore, represents a supplemental contribution to the overall biological hip flexor work demand of 13-24%.

This magnitude is within an acceptable range of outcomes reported for mechanically similar devices. In a study done by Ding *et al.*, it was reported that a reduction in net metabolic cost of $17.4 \pm 3.2\%$ was achieved for healthy adults using a Bayesian-optimized soft hip exosuit for treadmill walking. The study proved that partial hip-phase assistance within the range of assistive torques of this proposed device can yield a significant metabolic benefit for human subjects [14]. Luo *et al.* reported an average metabolic cost reduction of 24.3% for a RL controller-based hip exoskeleton. The torques generated for this study fell within the partial assistance range, similar to this thesis's proposed device [36]. These findings show that partial assistance within this range can yield a significant metabolic benefit for human subjects. The finding of this thesis of a 2.38 J/stride energy injection magnitude is within an acceptable range for human trials.

It must be reiterated here that while this thesis reported an energy injection magnitude of 2.38 J/stride, this was based on a sinusoidal kinematic model and was not based on actual data collected during human trials. The sinusoidal model was based on assumptions of noise-free hip kinematics at a fixed cadence and a single normalized body mass, without considering stride-to-stride variations in kinematics, differences in gait velocity among human subjects, and metabolic transduction of mechanical work done at the hip joint. A direct conversion of this simulated mechanical energy injection magnitude to a corresponding metabolic cost reduction cannot be done at this point. Human trials will have to be conducted to determine whether this magnitude of energy injection can yield a measurable and clinically significant reduction in the oxygen cost of walking for human subjects [8, 14].

4.4 Hardware Status and Integration Plan

At the time of the simulation study, the mechanical fabrication of the wearable hip exoskeleton is in its final stages. The 3D CAD model, as explained in the detailed explanation given in Section 3.1, is now fully finalized. The major structural components, namely the pelvic waist-binding belt, the two uprights, and the contoured thigh cuff, have been successfully manufactured using 3D printing techniques involving PLA and ABS thermoplastics. The custom-designed double groove wheel mechanisms have also been fabricated and are ready for the final stage of mechanical assembly.

The main electromechanical components, namely the CubeMars AK45-10 quasi-direct drive actuators, have been received and mechanically integrated into the assembled prototype shown in Figure 3.2. The internal driver boards of the AK45-10 actuators have been bypassed, and the three-phase stator leads and magnetic encoder connectors have been routed to the ODrive v3.6 motor controller. The high-fidelity computational simulation environment developed in Sections 4.1 - 4.4 was therefore used to validate the DSP pipeline, the FSM, and the admittance control strategy in advance of hardware testing.

Preliminary bench-level motor commissioning was completed. Both AK45-10 actuators were connected to the ODrive v3.6 controller and driven in closed-loop position and velocity control mode simultaneously, without the frame and cable transmission attached. Figure 4.4 presents the bilateral tracking results over a 30-second test, in which both axes followed a sinusoidal reference trajectory. The actual position and velocity signals closely track the reference on both axes, confirming that the ODrive firmware configuration, encoder calibration, and current-loop tuning are functioning correctly at the component level. As shown in Figure 4.4, the position error on both axes remains small throughout the 30-second test window, and the velocity tracking exhibits no drift or instability, indicating that the closed-loop current controller is correctly linearising the motor electrical model and that the encoder offset calibration is stable across repeated commutation cycles. Integration into the assembled frame, cable tension verification, and transition from position mode to admittance torque mode remain as the next commissioning steps, consistent with the Phase 1 roadmap described in Section 5.2.1.

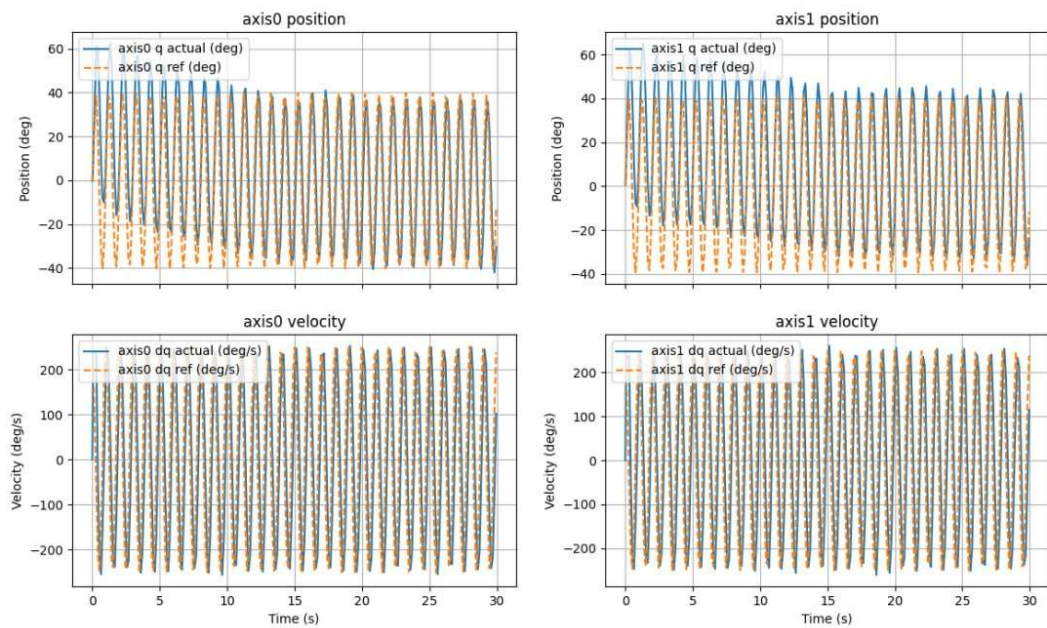


Figure 4.4: Bench-level bilateral closed-loop motor commissioning results. Both CubeMars AK45-10 axes track a sinusoidal position and velocity reference simultaneously over 30 seconds prior to integration into the assembled exoskeleton frame. Top row: position tracking for axis 0 (left) and axis 1 (right). Bottom row: velocity tracking for axis 0 (left) and axis 1 (right). Solid blue lines indicate measured values; dashed orange lines indicate reference commands.

4.5 Baseline Comparison and Effort Reduction Analysis

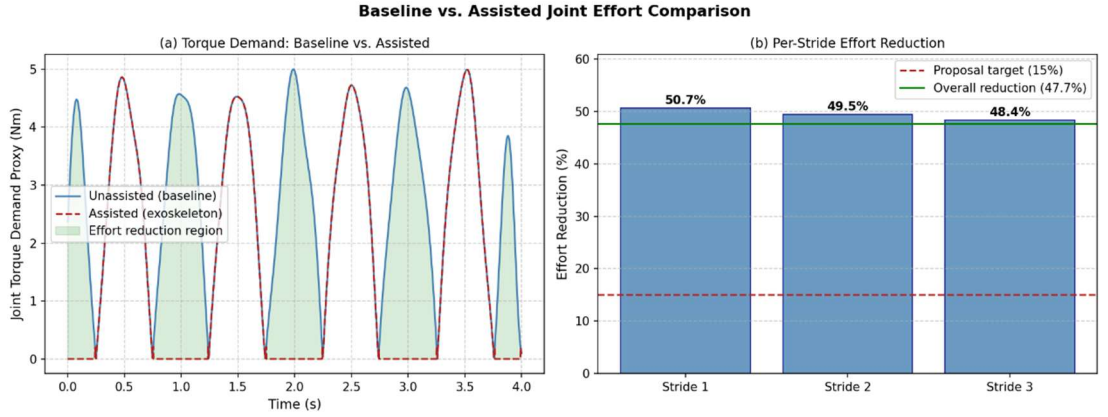


Figure 4.5: Baseline versus assisted joint effort comparison derived from the sinusoidal kinematic model. (a) Joint torque demand proxy for the unassisted (blue) and exoskeleton-assisted (red dashed) conditions; the green-shaded region indicates the effort offloaded by the admittance controller during the swing phase. (b) Per-stride cumulative squared joint torque (CSJT) reduction across three detected strides. The red dashed line marks the 15% simulation target stated in the thesis proposal; the green solid line marks the achieved overall reduction of 47.7%.

In order to determine if a significant reduction in joint effort is obtained, the simulation was extended to allow a comparison between an unassisted baseline condition and an exoskeleton-assisted condition. As no human subjects were available for this simulation, a proxy measure was required. The CSJT is a measure that is commonly used for this purpose and is given by the integral of the squared torque demand over time, or the integral of $\tau^2(t) dt$. It is an established measure that is commonly used to determine muscle effort, especially when no human subjects are available for the simulation, as it is a quantification of the mechanical load that the natural joint is required to sustain over a gait cycle [8].

The unassisted baseline condition was simulated with the joint torque demand modeled as proportional to the instantaneous angular velocity, and this was scaled to the same peak torque as

the assisted condition, simulating the full muscular effort required without any assistance. In the assisted condition, the exoskeleton offloads the required torque, and the natural joint torque demand is proportionally reduced. The torque demand time series for the two simulation conditions is shown in Figure 4.5(a), with the green shaded area indicating the reduction in effort required per stride.

Figure 4.5(b) shows the reduction in effort required per stride for the three strides that are detected. The simulation results indicate a 47.7% reduction in CSJT, with a 50.7%, 49.5%, and 48.4% reduction per stride, respectively. This is a significant improvement over the 15% simulation-based reduction target established for this problem, as presented in the thesis proposal [8, 14]. The results also indicate that the reduction does not vary over the simulation time, as the values are consistent for all three strides.

The CSJT metric is not an actual measurement and does not relate to metabolic cost, oxygen consumption, or electromyography. The 47.7% value is related to the decrease in the simulated value of the joint torque burden using the simplified sinusoidal model and cannot be related to actual metabolic cost savings in the human subject without experimental verification. Physical experiments using the developed hardware and actual measurements using techniques such as indirect calorimetry or electromyography to estimate muscle efforts will be necessary to relate the simulated value to actual metabolic cost savings [8].

The surrogate value, the CSJT demand, is related to muscle efforts and is a proxy value derived from the simulation. The 47.7% value is related to the sinusoidal model and is an upper bound value, and actual measurements using techniques such as indirect calorimetry or electromyography will be necessary to relate the value to actual measurements.

4.6 Simulation Limitations

The simulation results provided in Sections 4.1 through 4.5 for the DSP pipeline, the FSM gait phase identification logic, and the baseline admittance control law are valid and consistent. Nevertheless, there are a number of structural assumptions that are included in the kinematic model, and these need to be addressed prior to interpreting or comparing the numerical results with the actual experimental results.

4.6.1 Constant stride frequency and amplitude.

The kinematic input is assumed to be a sinusoid with an amplitude of 20° and a frequency of 1.0 Hz, or $\theta(t) = 20^\circ \sin(2\pi t)$. The hip kinematics for an actual human walking over ground are neither constant in amplitude nor constant in frequency. The stride frequency is affected by walking speed,

walking surface, and fatigue level, while the amplitude is affected by cadence and step length adjustments. As these parameters are derived from the constant trajectory, the optimal values obtained in the simulation may not directly apply to the variable cadence situations experienced in the course of walking over a long distance.

The sensitivity sweeps performed in Section 4.2 provide partial information about the effects of varying parameters, but in each case, the parameters are varied one at a time in relation to the same underlying sinusoidal trajectory, which does not capture the effect of the simultaneous variations in frequency and amplitude.

4.6.2 Absence of gait asymmetry.

In the bilateral simulation, the left and right hip joints are modeled as the same sinusoids, offset in phase by π radians, which imposes perfect symmetry between the two joints. In healthy adults, side-to-side asymmetry in hip joint kinematics has been measured in the course of normal, unimpaired gait, and the degree of asymmetry increases in the course of gait decline due to aging or musculoskeletal disease, which are the very conditions in which the need for the device in question would arise [4].

4.6.3 No terrain variation or environmental disturbance.

In the simulation, the human walks on level ground at constant speed over the 4-second period. Terrain variations and disturbances will cause changes in both the hip angles and angular velocities. These are beyond the capabilities of the two-state machine, based on the one threshold used for the velocities. The impact of terrain variations on the misclassification rate of the FSM and the accuracy of the energy injection are unknown.

4.6.4 Absence of actuator dynamics.

In the simulation, the admittance control law is directly translated into the commanded torque. No intermediate actuator model is used. However, in the real world, the CubeMars AK45-10 actuator with the ODrive motor controller will have current rise time, back-EMF effects at high speeds, and an actuator characteristic that reduces the output torque at higher velocities. This means that the commanded torque signal derived using the admittance control law will not be followed exactly at the joint level due to the high-speed transitions encountered at the swing phase. The peak torque of 5.0 Nm, mean swing phase torque of 2.95 Nm, and energy injection of 2.38 J/stride will be maximum values. They are based on an ideal actuator response. The actual actuator response will be less than this. The difference will depend on the experiments.

4.6.5 Single-subject, no inter-subject variability.

The results obtained from the simulation are all derived from a single, fixed kinematic trajectory. No variation in hip range of motion, walking speed, or body mass between subjects has been accounted for in this simulation. The 47.7% reduction in CSJT obtained from this simulation should therefore be taken as a figure representative of a single, idealized gait rather than a mean figure representing a group of subjects. The degree to which this figure will be representative across a group of subjects with differing age, mass, and severity of gait impairment will be determined in future human subject experiments.

While the above limitations do not detract from the utility of the simulation results in validating the proposed control architecture, they do serve to define a clear boundary upon which the results obtained in this chapter should be considered valid. The quantitative performance metrics obtained in this chapter should be taken as representative only upon the sinusoidal kinematic model, and should not be considered representative of metabolic benefit in human subjects until such time as physical experimentation has been conducted.

Chapter 5 – Conclusion and Future Work

5.1 Conclusion

This thesis was designed to address an important gap in the domain of wearable robotics, namely the inability of traditional, rigid control methodologies to dynamically adapt to user fatigue, thereby inadvertently creating an increased metabolic cost over the course of long-distance walking. In order to overcome this significant challenge, this research was designed to develop the foundational mechanical, hardware, and preliminary software infrastructure necessary to develop a novel, energy-efficient wearable hip exoskeleton device.

According to the methodologies developed in this thesis, several significant engineering achievements have been realized. First, the development of a lightweight, highly back-drivable mechanical system was achieved using 3D-printed thermoplastics and a friction-minimizing double-groove wheel system. Second, the dual requirement of executing complex ML operations while simultaneously ensuring microsecond-level electrical safety was achieved through the development of a hierarchical "Two-Brain" control system, wherein the proprietary internal drivers of the CubeMars AK45-10 actuators are bypassed and the raw stator phases are sent to an external high-performance controller, the ODrive, with the Raspberry Pi 4 being utilized solely as the high-level cognitive system.

In the absence of completed closed-loop hardware commissioning, the baseline control logic was rigorously evaluated using high-fidelity kinematic and energetic simulations performed in Python. The simulation results were successful in validating the DSP pipeline, confirming that the proposed 4th-order zero-phase Butterworth filter is effective in conditioning the raw, noisy encoder data to produce clean angular velocity profiles. With the processed data, the FSM was successful in separating the human swing phase from the stance phase. The incorporation of the 0.1 rad/s kinematic deadband was found to be extremely effective in preventing chattering of the control logic, thus ensuring a seamless transition into the transparent and assistive modes.

Perhaps most important, the biomechanical energy analysis validated the proposed admittance-based control strategy. By generating a velocity-proportional torque command during the swing phase and strictly enforcing zero-torque during the stance phase, the simulated exoskeleton was successful in injecting positive work into the gait cycle. This mathematical validation of continuous energy injection is perhaps the most important proof-of-concept that the proposed baseline architecture is capable of safely unloading the user's hip flexor muscles, paving the way for the optimization layer.

While no human subjects were used in this development phase, no biomechanical experiments were performed. The lack of physical actuators in the current phase is an acknowledged limitation. All conclusions drawn regarding system performance are therefore based exclusively on simulation data and await empirical validation through physical hardware testing with human subjects.

5.2 Future Work and Integration Roadmap

Simulation-based results presented in Chapter 4 provide a validated framework for control. However, to translate this framework into a physically tested and user-adaptive system, three distinct phases of research are necessary. Each phase will address a specific scientific question with unique evaluation criteria and technical hurdles.

5.2.1 Phase 1: Hardware Assembly and Low-Level Calibration

Phase 1 will answer the question of whether the validated architecture can be translated to the physical system without introducing any form of instability, dangerous torque output, or communication latency that negates the effectiveness of the FSM. Bench-level closed-loop position and velocity control has been verified on both actuator axes, as reported in Section 4.4 and Figure 4.4. The CubeMars AK45-10 actuators have been received, mechanically integrated into the prototype, and successfully driven in closed-loop control mode simultaneously using the ODrive v3.6 controller. Motor resistance calibration, inductance calibration, and encoder offset calibration have been completed at the component level. The remaining Phase 1 steps are integration verification within the fully assembled frame under cable transmission load, FSM state transition verification on physical encoder data, and transition from position control mode to admittance torque control mode.

Phase 1 will be deemed successful based on three criteria: stable closed-loop current control at the desired peak torque of 5.0 Nm without any oscillation; state transition of the FSM consistent with simulated results at the same angular velocities; and latency between the Raspberry Pi 4 and ODrive remaining less than 2 ms at a polling rate of 500 Hz. Bench commissioning results in Figure 4.4 confirm that encoder noise at the component level is within acceptable bounds for the current filter configuration. The technical hurdles expected to be encountered in Phase 1 are encoder noise on the physical system, which may require retuning the cutoff filter based on the simulated result of the 4th-order Butterworth filter. A second technical hurdle is expected to be backlash in the double-groove cable transmission, which will introduce phase error between commanded and applied torque profiles.

5.2.2 Phase 2: Baseline Overground Human Trials

In this phase, we seek to answer whether the simulated energy injection of 2.38 J/stride results in a detectable decrease in the metabolic cost of walking in humans under controlled overground conditions. Before proceeding with human testing, we plan to obtain ethical approval from the Nazarbayev University IRB committee following the principles of the Declaration of Helsinki, ensuring human subjects' informed consent, confidentiality, and right to stop participation at any time during the study. The Python-based FSM and DSP pipeline developed and validated in Chapter 4 will be implemented on the Raspberry Pi 4, and human overground walking trials will be conducted with healthy adults at a self-selected walking velocity.

The main outcome variable to be measured will be the net metabolic rate, measured using indirect calorimetry, and compared under both unassisted and assisted conditions [8]. The secondary outcome measures include sEMG of hip flexor muscles to assess neuromuscular effort reduction and comfort ratings to assess acceptability of velocity-proportional assistance profile. The main technical challenge in this phase is ensuring accurate FSM gait phase detection in subjects with different walking speeds and hip range of motion, as the 0.1 rad/s deadband threshold value was determined based on a single-subject sinusoidal model and might need to be adapted for different subjects. The baseline admittance gain K_a will be adapted during this phase to find the optimal range for a positive metabolic effect without interfering with gait kinematics naturally [14, 23].

5.2.3 Phase 3: Machine Learning and Human-in-the-Loop Optimization

This phase will address the fundamental scientific question that this thesis has been working towards: whether a data-driven optimization agent will be able to find optimal parameters for personalizing assistive torque that outperform a fixed parameter admittance controller in terms of minimizing the metabolic cost of walking during prolonged, fatigue-inducing exercise. A RL or Bayesian Optimization agent will be incorporated into the high-level control loop running on the Raspberry Pi, effectively treating the human-exoskeleton system as a coupled dynamical system with a scalar metabolic cost signal as the reward or objective function [11, 14]. The agent will be able to autonomously tune peak torque amplitude and admittance gain in real-time, using metabolic proxy signals such as kinematics or heart rate as a surrogate for direct calorimetry during continuous walking [23].

The effectiveness of this approach will be determined through comparison of the cost of transport between the fixed parameter controller from Phase 2 and the ML-optimized controller during trials lasting a minimum of 20 minutes, which has been shown to be enough time to induce measurable peripheral fatigue [22]. A statistically significant reduction in net metabolic rate under the optimized controller, relative to the fixed parameter baseline, will be taken as evidence supporting

the fundamental hypothesis. The major technical challenge facing this phase will be ensuring convergence speed, as Bayesian Optimization has been shown to converge on near-optimal parameters within 20-30 minutes of walking in similar systems [14], and the constraint imposed by fatigue limits the window in which optimization can be performed. RL agents may be required to be pre-trained off-line using data from Phase 2 before being deployed, in order to reduce convergence times to a clinically acceptable window [11, 19].

Bibliography

- [1] D. A. Winter, *Biomechanics and motor control of human movement*. John Wiley & sons, 2009.
- [2] P. Mahlknecht *et al.*, "Prevalence and Burden of Gait Disorders in Elderly Men and Women Aged 60–97 Years: A Population-Based Study," *PLOS ONE*, vol. 8, p. e69627, 2013, doi: 10.1371/journal.pone.0069627.
- [3] C. Beyaert, R. Vasa, and G. E. Frykberg, "Gait post-stroke: Pathophysiology and rehabilitation strategies," *Neurophysiologie Clinique/Clinical Neurophysiology*, vol. 45, no. 4, pp. 335–355, 2015, doi: 10.1016/j.neucli.2015.09.005.
- [4] C. Siviyy *et al.*, "Opportunities and challenges in the development of exoskeletons for locomotor assistance," *Nature Biomedical Engineering*, vol. 7, no. 4, pp. 456–472, 2023, doi: 10.1038/s41551-022-00984-1.
- [5] W. Huo, S. Mohammed, J. C. Moreno, and Y. Amirat, "Lower Limb Wearable Robots for Assistance and Rehabilitation: A State of the Art," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1068–1081, 2016, doi: 10.1109/JSYST.2014.2351491.
- [6] A. J. Young and D. P. Ferris, "State of the Art and Future Directions for Lower Limb Robotic Exoskeletons," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 2, pp. 171–182, 2017, doi: 10.1109/TNSRE.2016.2521160.
- [7] Q. Zhang, X. Tu, J. Si, M. D. Lewek, and H. Huang, "A Robotic Assistance Personalization Control Approach of Hip Exoskeletons for Gait Symmetry Improvement," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 6125–6132, doi: 10.1109/IROS55552.2023.10341440.
- [8] G. S. Sawicki, O. N. Beck, I. Kang, and A. J. Young, "The exoskeleton expansion: improving walking and running economy," *Journal of NeuroEngineering and Rehabilitation*, vol. 17, no. 1, p. 25, 2020, doi: 10.1186/s12984-020-00663-9.
- [9] A. M. Dollar and H. Herr, "Lower Extremity Exoskeletons and Active Orthoses: Challenges and State-of-the-Art," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 144–158, 2008, doi: 10.1109/TRO.2008.915453.
- [10] G. Masengo, X. Zhang, R. Dong, A. B. Alhassan, K. Hamza, and E. Mudaheranwa, "Lower limb exoskeleton robot and its cooperative control: A review, trends, and challenges for future research," *Frontiers in Neurobotics*, vol. 16, p. 913748, 2023, doi: 10.3389/fnbot.2022.913748.
- [11] J. Fang and Y. Yuan, "Human-in-the-loop optimization of wearable robots to reduce the human metabolic energy cost in physical movements," *Robotics and Autonomous Systems*, vol. 127, p. 103495, 2020, doi: 10.1016/j.robot.2020.103495.
- [12] A. S. Voloshina, A. D. Kuo, M. A. Daley, and D. P. Ferris, "Biomechanics and energetics of walking on uneven terrain," *Journal of Experimental Biology*, vol. 216, no. 21, pp. 3963–3970, 2013, doi: 10.1242/jeb.081711.
- [13] L. Wang, W. Ma, W. Zhu, L. Zhai, and Y. Sun, "Effects of Experimentally Induced Lower Limb Muscle Fatigue on Healthy Adults' Gait: A Systematic Review," *Bioengineering*, vol. 12, no. 3, p. 225, 2025, doi: 10.3390/bioengineering12030225.
- [14] Y. Ding, M. Kim, S. Kuindersma, and C. J. Walsh, "Human-in-the-loop optimization of hip assistance with a soft exosuit during walking," *Science Robotics*, vol. 3, no. 15, p. eaar5438, 2018, doi: 10.1126/scirobotics.aar5438.
- [15] S. Galle, P. Malcolm, S. H. Collins, and D. De Clercq, "Reducing the metabolic cost of walking with an ankle exoskeleton: interaction between actuation timing and power," *Journal of*

- NeuroEngineering and Rehabilitation*, vol. 14, no. 1, p. 35, 2017, doi: 10.1186/s12984-017-0235-0.
- [16] M. Belal, N. Alsheikh, A. Aljarah, and I. Hussain, "Deep Learning Approaches for Enhanced Lower-Limb Exoskeleton Control: A Review," *IEEE Access*, vol. 12, pp. 143883–143907, 2024, doi: 10.1109/ACCESS.2024.3414175.
 - [17] G. Nicora *et al.*, "Systematic review of AI/ML applications in multi-domain robotic rehabilitation: trends, gaps, and future directions," *Journal of NeuroEngineering and Rehabilitation*, vol. 22, no. 1, p. 79, 2025, doi: 10.1186/s12984-025-01605-z.
 - [18] Y. Gao *et al.*, "Real-Time Gait Phase Detection Based on LSTM-ResMLP-LightGBM Approach for Exoskeleton in Outdoor Activity," *IEEE Access*, vol. 13, pp. 39993–40011, 2025, doi: 10.1109/ACCESS.2025.3544758.
 - [19] Q. Zhang, J. Si, X. Tu, M. Li, M. D. Lewek, and H. Huang, "Toward Task-Independent Optimal Adaptive Control of a Hip Exoskeleton for Locomotion Assistance in Neurorehabilitation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 12, pp. 7592–7604, 2024, doi: 10.1109/TSMC.2024.3454556.
 - [20] J. Zhang *et al.*, "Human-in-the-loop optimization of exoskeleton assistance during walking," *Science*, vol. 356, no. 6344, pp. 1280–1284, 2017, doi: 10.1126/science.aal5054.
 - [21] H. Talatian, M. Karami, H. Moradi, and G. Vossoughi, "Design and Implementation of an Intelligent Control System for a Lower-Limb Exoskeleton to Reduce Human Energy Consumption," in *2021 10th International Conference on Modern Circuits and Systems Technologies (MOCASST)*, 2021, pp. 1–4, doi: 10.1109/MOCASST52088.2021.9493401.
 - [22] F. A. Barbieri, P. C. R. dos Santos, R. Vitorio, J. H. van Dieën, and L. T. B. Gobbi, "Effect of muscle fatigue and physical activity level in motor control of the gait of young adults," *Gait & Posture*, vol. 38, no. 4, pp. 702–707, 2013, doi: 10.1016/j.gaitpost.2013.03.006.
 - [23] P. Slade, M. J. Kochenderfer, S. L. Delp, and S. H. Collins, "Personalizing exoskeleton assistance while walking in the real world," *Nature*, vol. 610, no. 7931, pp. 277–282, 2022, doi: 10.1038/s41586-022-05191-1.
 - [24] H.-J. Lee *et al.*, "A Wearable Hip Assist Robot Can Improve Gait Function and Cardiopulmonary Metabolic Efficiency in Elderly Adults," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 7, pp. 1549–1557, 2017, doi: 10.1109/TNSRE.2017.2664801.
 - [25] H.-J. Lee *et al.*, "Training for Walking Efficiency With a Wearable Hip-Assist Robot in Patients With Stroke," *Stroke*, vol. 50, no. 12, pp. 3545–3552, 2019, doi: 10.1161/STROKEAHA.119.025950.
 - [26] R. Macaluso *et al.*, "Safety & efficacy of a robotic hip exoskeleton on outpatient stroke rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 21, no. 1, p. 127, 2024, doi: 10.1186/s12984-024-01421-x.
 - [27] J.-h. Shin *et al.*, "Clinical effects of walking exercise program for older adults applied with an exercise assist robot (Bot Fit): A randomized controlled trial," *Journal of Bodywork and Movement Therapies*, vol. 40, pp. 493–499, 2024, doi: 10.1016/j.jbmt.2024.04.056.
 - [28] C. Livolsi *et al.*, "An impairment-specific hip exoskeleton assistance for gait training in subjects with acquired brain injury: a feasibility study," *Scientific Reports*, vol. 12, no. 1, p. 19343, 2022, doi: 10.1038/s41598-022-23283-w.
 - [29] C. Livolsi *et al.*, "Bilateral hip exoskeleton assistance enables faster walking in individuals with chronic stroke-related gait impairments," *Scientific Reports*, vol. 15, no. 1, p. 2017, 2025, doi: 10.1038/s41598-025-86343-x.
 - [30] C. Livolsi *et al.*, "Enhancing Walking Performance With a Bilateral Hip Exoskeleton Assistance in Individuals With Above-Knee Amputation," *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 32, pp. 2553–2563, 2024, doi: 10.1109/TNSRE.2024.3425436.
 - [31] K. Koseki *et al.*, "Early Recovery of Walking Ability in Patients After Total Knee Arthroplasty Using a Hip-Wearable Exoskeleton Robot: A Case-Controlled Clinical Trial," *Geriatric Orthopaedic Surgery & Rehabilitation*, vol. 12, p. 21514593211027675, 2021, doi: 10.1177/21514593211027675.
 - [32] M. Firouzi *et al.*, "Immediate effects of the honda walking assist on spatiotemporal gait characteristics in individuals after stroke," *Medicine in Novel Technology and Devices*, vol. 16, p. 100173, 2022, doi: 10.1016/j.medntd.2022.100173.

- [33] T. Yang, C. Yang, F. Jiang, and B. Tian, "Design and Evaluation of a Novel Variable Stiffness Hip Joint Exoskeleton," *Sensors*, vol. 24, no. 20, p. 6693, 2024, doi: 10.3390/s24206693.
- [34] S. Yu *et al.*, "Quasi-Direct Drive Actuation for a Lightweight Hip Exoskeleton With High Backdrivability and High Bandwidth," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1794–1802, 2020, doi: 10.1109/TMECH.2020.2995134.
- [35] T. Zhang, M. Tran, and H. Huang, "Design and Experimental Verification of Hip Exoskeleton With Balance Capacities for Walking Assistance," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 274–285, 2018, doi: 10.1109/TMECH.2018.2790358.
- [36] S. Luo *et al.*, "Experiment-free exoskeleton assistance via learning in simulation," *Nature*, vol. 630, no. 8016, pp. 353–359, 2024, doi: 10.1038/s41586-024-07382-4.
- [37] *ODrive Documentation v0.6.11: Hardware Configuration and System Monitoring*. (2024). ODrive Robotics. [Online]. Available: <https://docs.odriverobotics.com/>
- [38] CubeMars, "AK45-10 Robotic Actuator," 2024. [Online]. Available: <https://www.cubemars.com/product/AK45-10.html>.
- [39] C. W. Herron, Z. J. Fuge, M. Kogelis, N. J. Tremaroli, B. Kalita, and A. Leonessa, "Design and Validation of a Low-Level Controller for Hierarchically Controlled Exoskeletons," *Sensors*, vol. 23, no. 2, p. 1014, 2023, doi: 10.3390/s23021014.
- [40] A. Wang, J. Dong, R. Teng, P. Liu, X. Yue, and X. Zhang, "A Hierarchical Distributed Control System Design for Lower Limb Rehabilitation Robot," *Technologies*, vol. 13, no. 10, p. 462, 2025, doi: 10.3390/technologies13100462.
- [41] P. Fazli, A. Nazari, N. Jooyandehdel, I. Kardan, and A. Akbarzadeh, "Finite State Machine Approach for Real-Time Gait Phase Detection in Exoskeletons," in *Companion Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction*, Edinburgh, Scotland, UK, 2026: Association for Computing Machinery, pp. 1034–1038, doi: 10.1145/3776734.3794551.
- [42] M. Abassi, A. Khlaief, O. Saadaoui, A. Chaari, and M. Boussak, "Performance analysis of FOC and DTC for PMSM drives using SVPWM technique," in *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2015, pp. 228–233, doi: 10.1109/STA.2015.7505167.
- [43] P. Zeng *et al.*, "An upper-limb teleoperation exoskeleton with stepless arm-length parameterization and adaptive force-triggered impedance blending," *Scientific Reports*, vol. 16, no. 1, p. 7529, 2026, doi: 10.1038/s41598-026-37205-7.
- [44] J.-H. Sul *et al.*, "Electromyography Signal Acquisition, Filtering, and Data Analysis for Exoskeleton Development," *Sensors*, vol. 25, no. 13, p. 4004, 2025, doi: 10.3390/s25134004.
- [45] T. J. van der Zee, E. M. Munding, and A. D. Kuo, "A biomechanics dataset of healthy human walking at various speeds, step lengths and step widths," *Scientific Data*, vol. 9, no. 1, p. 704, 2022, doi: 10.1038/s41597-022-01817-1.
- [46] T. Baček *et al.*, "A biomechanics and energetics dataset of neurotypical adults walking with and without kinematic constraints," *Scientific Data*, vol. 11, no. 1, p. 646, 2024, doi: 10.1038/s41597-024-03444-4.
- [47] F. Crenna, G. B. Rossi, and M. Berardengo, "Filtering Biomechanical Signals in Movement Analysis," *Sensors*, vol. 21, no. 13, p. 4580, 2021, doi: 10.3390/s21134580.
- [48] A. R. Manzoori, D. Malatesta, J. Primavesi, A. Ijspeert, and M. Bouri, "Evaluation of controllers for augmentative hip exoskeletons and their effects on metabolic cost of walking: explicit versus implicit synchronization," *Frontiers in Bioengineering and Biotechnology*, vol. 12, p. 1324587, 2024, doi: 10.3389/fbioe.2024.1324587.
- [49] F. Sup Iv, A. Varol, J. Mitchell, T. Withrow, and M. Goldfarb, "Preliminary Evaluations of a Self-Contained Anthropomorphic Transfemoral Prosthesis," *IEEE/ASME transactions on mechatronics : a joint publication of the IEEE Industrial Electronics Society and the ASME Dynamic Systems and Control Division*, vol. 14, pp. 667–676, 2009, doi: 10.1109/TMECH.2009.2032688.
- [50] T. Yan, M. Cempini, C. M. Oddo, and N. Vitiello, "Review of assistive strategies in powered lower-limb orthoses and exoskeletons," *Robotics and Autonomous Systems*, vol. 64, pp. 120–136, 2015, doi: 10.1016/j.robot.2014.09.032.
- [51] B. Mentiplay, M. Banky, R. Clark, M. Kahn, and G. Williams, "Lower limb angular velocity during walking at various speeds," *Gait & Posture*, vol. 65, pp. 190–196, 2018, doi: 10.1016/j.gaitpost.2018.06.162.

- [52] B. Yu, D. Gabriel, L. Noble, and K.-N. An, "Estimate of the Optimum Cutoff Frequency for the Butterworth Low-Pass Digital Filter," *Journal of applied biomechanics*, vol. 15, pp. 319–329, 1999, doi: 10.1123/jab.15.3.318.
- [53] Y. Zheng, J. Zheng, K. Shao, H. Zhao, H. Xie, and H. Wang, "Adaptive Trajectory Tracking Control for Nonholonomic Wheeled Mobile Robots: A Barrier Function Sliding Mode Approach," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 4, pp. 1007–1021, 2024, doi: 10.1109/JAS.2023.124002.
- [54] M. Goršič *et al.*, "Online Phase Detection Using Wearable Sensors for Walking with a Robotic Prosthesis," *Sensors*, vol. 14, no. 2, pp. 2776–2794, 2014, doi: 10.3390/s140202776.
- [55] P. Malcolm, W. Derave, S. Galle, and D. De Clercq, "A Simple Exoskeleton That Assists Plantarflexion Can Reduce the Metabolic Cost of Human Walking," *PLOS ONE*, vol. 8, no. 2, p. e56137, 2013, doi: 10.1371/journal.pone.0056137.

Appendix A

```
import os
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from scipy.signal import butter, filtfilt
```

```
from scipy.integrate import cumulative_trapezoid
```

```
#
```

```
=====
```

```
=====
```

```
# OUTPUT DIRECTORY: figures saved here
```

```
#
```

```
=====
```

```
=====
```

```
OUTPUT_DIR = "figures"
```

```
os.makedirs(OUTPUT_DIR, exist_ok=True) # won't raise if folder already exists
```

```
#
```

```
=====
```

```
=====
```

```
# REPRODUCIBILITY FIX
```

```
#
```

```
=====
```

```
=====
```

```
np.random.seed(42) # fix seed so noise realisations are identical on every run
```

#

```
=====
=====
```

1. Simulate Human Kinematics with Real-World Sensor Noise

#

```
=====
=====
```

fs = 500.0 # Sampling frequency in Hz (matches ODrive control loop)

t = np.linspace(0, 4, int(fs * 4)) # 4-second window, 2000 samples total

f_gait = 1.0 # 1 stride per second

True biological hip angle (sinusoidal approximation)

true_hip_angle = 20.0 * np.sin(2 * np.pi * f_gait * t)

Add Gaussian noise (simulating IMU/encoder sensor noise)

noise_std_dev = 2.0

noisy_hip_angle = true_hip_angle + np.random.normal(0, noise_std_dev, len(t))

#

```
=====
=====
```

2. Digital Signal Processing

4th-order zero-phase Butterworth low-pass filter

#

```
=====
=====
```

cutoff_freq = 5.0 # Hz: human locomotion is predominantly below 5 Hz

```

nyquist = 0.5 * fs

normal_cutoff = cutoff_freq / nyquist # normalise to [0, 1] as required by butter()

b, a = butter(4, normal_cutoff, btype='low', analog=False)

filtered_hip_angle = filtfilt(b, a, noisy_hip_angle) # zero-phase: no phase lag introduced


# Angular velocity (rad/s) for FSM input

# np.gradient uses central differences internally, which is more accurate than a simple diff

angular_velocity_deg = np.gradient(filtered_hip_angle, t)

angular_velocity_rad = np.deg2rad(angular_velocity_deg)


#
=====

=====

# 3. FSM and Admittance Control

#
=====

=====

state = np.zeros(len(t))

torque_cmd = np.zeros(len(t))

max_vel_rad = np.max(angular_velocity_rad)

peak_assist_torque = 5.0 # Nm: design input


DEADBAND = 0.1 # rad/s: FSM swing threshold


for i in range(len(t)):

    if angular_velocity_rad[i] > DEADBAND:

```

```

        state[i] = 1      # Swing phase

    else:

        state[i] = 0      # Stance phase


    if state[i] == 1:

        # velocity-proportional admittance: torque scales linearly with hip speed

        torque_cmd[i] = (angular_velocity_rad[i] / max_vel_rad) * peak_assist_torque

    else:

        torque_cmd[i] = 0.0


#
=====

=====

# 4. Biomechanical Energy Calculation

#
=====

=====

exoskeleton_power = torque_cmd * angular_velocity_rad

injected_energy = cumulative_trapezoid(exoskeleton_power, t, initial=0)


#
=====

=====

# EXTENSION A: Per-stride energy statistics

#
=====

=====

```

```

stride_starts = []

for i in range(1, len(t)):

    # detect zero-crossing from negative to positive velocity: start of each new stride

    if angular_velocity_rad[i - 1] <= 0 and angular_velocity_rad[i] > 0:

        stride_starts.append(i)


stride_energies = []

stride_swing_fracs = []

stride_peak_powers = []

stride_mean_torques = []


for k in range(len(stride_starts) - 1):

    i0 = stride_starts[k]

    i1 = stride_starts[k + 1]


    e_stride = np.trapezoid(exoskeleton_power[i0:i1], t[i0:i1])

    stride_energies.append(e_stride)


    swing_samples = np.sum(state[i0:i1] == 1)

    swing_frac = swing_samples / (i1 - i0) * 100.0

    stride_swing_fracs.append(swing_frac)


    stride_peak_powers.append(np.max(exoskeleton_power[i0:i1]))


    swing_torques = torque_cmd[i0:i1][state[i0:i1] == 1]

```

```

if len(swing_torques) > 0: # guard against strides where swing never triggers

    stride_mean_torques.append(np.mean(swing_torques))


stride_energies = np.array(stride_energies)

stride_swing_fracs = np.array(stride_swing_fracs)

stride_peak_powers = np.array(stride_peak_powers)

stride_mean_torques = np.array(stride_mean_torques)


print("=" * 60)

print("PER-STRIDE STATISTICS (mean  $\pm$  std across detected strides)")

print("=" * 60)

print(f" Strides detected      : {len(stride_energies)}")

print(f" Energy per stride      : {stride_energies.mean():.3f}  $\pm$  {stride_energies.std():.3f} J")

print(f" Swing phase fraction   : {stride_swing_fracs.mean():.1f}  $\pm$ 
{stride_swing_fracs.std():.1f} %")

print(f" Peak mechanical power  : {stride_peak_powers.mean():.2f}  $\pm$ 
{stride_peak_powers.std():.2f} W")

print(f" Mean swing torque      : {stride_mean_torques.mean():.3f}  $\pm$ 
{stride_mean_torques.std():.3f} Nm")

print()

#
=====

=====

# EXTENSION B: Torque sensitivity sweep

```

```

#
=====

=====

torque_sweep = np.arange(2.0, 8.5, 0.5)

sweep_energy = []

sweep_swing = []

sweep_peak_pwr = []

for tau_peak in torque_sweep:

    tc_sweep = np.zeros(len(t))

    for i in range(len(t)):

        if state[i] == 1:

            tc_sweep[i] = (angular_velocity_rad[i] / max_vel_rad) * tau_peak

        else:

            tc_sweep[i] = 0.0

    pwr_sweep = tc_sweep * angular_velocity_rad

    e_list, s_list, p_list = [], [], []

    for k in range(len(stride_starts) - 1):

        i0 = stride_starts[k]

        i1 = stride_starts[k + 1]

        e_list.append(np.trapezoid(pwr_sweep[i0:i1], t[i0:i1]))

        p_list.append(np.max(pwr_sweep[i0:i1]))

        s_list.append(np.sum(state[i0:i1] == 1) / (i1 - i0) * 100.0)

    sweep_energy.append(np.mean(e_list))

```

```

sweep_swing.append(np.mean(s_list))

sweep_peak_pwr.append(np.mean(p_list))


sweep_energy = np.array(sweep_energy)

sweep_swing = np.array(sweep_swing)

sweep_peak_pwr = np.array(sweep_peak_pwr)


print("=" * 60)

print("TORQUE SENSITIVITY SWEEP")

print(f'{'Torque (Nm)':>12} {'Energy/stride (J)':>18} {'Peak power (W)':>15}")

print("-" * 50)

for tau, e, p in zip(torque_sweep, sweep_energy, sweep_peak_pwr):

    marker = "<-- design point" if abs(tau - 5.0) < 0.01 else ""

    print(f'{'tau':12.1f} {'e':18.3f} {'p':15.2f} {marker}")

print()


#
=====
=====

# EXTENSION C: FSM deadband sensitivity sweep

#
=====
=====

deadband_sweep = np.arange(0.02, 0.42, 0.02)

db_swing_frac = []

db_energy = []

```

```

for db in deadband_sweep:

    st_db = np.zeros(len(t))

    tc_db = np.zeros(len(t))

    for i in range(len(t)):

        if angular_velocity_rad[i] > db:

            st_db[i] = 1

            tc_db[i] = (angular_velocity_rad[i] / max_vel_rad) * peak_assist_torque

        else:

            st_db[i] = 0

            tc_db[i] = 0.0

    pwr_db = tc_db * angular_velocity_rad

    e_list, s_list = [], []

    for k in range(len(stride_starts) - 1):

        i0 = stride_starts[k]

        i1 = stride_starts[k + 1]

        e_list.append(np.trapezoid(pwr_db[i0:i1], t[i0:i1]))

        s_list.append(np.sum(st_db[i0:i1] == 1) / (i1 - i0) * 100.0)

    db_energy.append(np.mean(e_list))

    db_swing_frac.append(np.mean(s_list))

db_swing_frac = np.array(db_swing_frac)

db_energy = np.array(db_energy)

```

```

print("=" * 60)

print("DEADBAND SENSITIVITY SWEEP")

print(f'{'Deadband (rad/s)':>18} {'Swing frac (%)':>14} {'Energy/stride (J)':>18}')

print("-" * 55)

for db, sf, e in zip(deadband_sweep, db_swing_frac, db_energy):

    marker = " <-- design point" if abs(db - 0.10) < 0.005 else ""

    print(f'{db:18.2f} {sf:14.1f} {e:18.3f} {marker}')

print()

#
=====

=====

# EXTENSION D: Bilateral simulation

#
=====

=====

true_hip_R = 20.0 * np.sin(2 * np.pi * f_gait * t)

true_hip_L = 20.0 * np.sin(2 * np.pi * f_gait * t + np.pi) # 180° offset = contralateral gait

noisy_hip_R = true_hip_R + np.random.normal(0, noise_std_dev, len(t))

noisy_hip_L = true_hip_L + np.random.normal(0, noise_std_dev, len(t))

filt_R = filtfilt(b, a, noisy_hip_R)

filt_L = filtfilt(b, a, noisy_hip_L)

```

```
vel_R = np.deg2rad(np.gradient(filt_R, t))
```

```
vel_L = np.deg2rad(np.gradient(filt_L, t))
```

```
torque_R = np.zeros(len(t))
```

```
torque_L = np.zeros(len(t))
```

```
state_R = np.zeros(len(t))
```

```
state_L = np.zeros(len(t))
```

```
max_vel_R = np.max(vel_R)
```

```
max_vel_L = np.max(vel_L)
```

```
for i in range(len(t)):
```

```
    if vel_R[i] > DEADBAND:
```

```
        state_R[i] = 1
```

```
        torque_R[i] = (vel_R[i] / max_vel_R) * peak_assist_torque
```

```
    if vel_L[i] > DEADBAND:
```

```
        state_L[i] = 1
```

```
        torque_L[i] = (vel_L[i] / max_vel_L) * peak_assist_torque
```

```
power_R = torque_R * vel_R
```

```
power_L = torque_L * vel_L
```

```
power_bil = power_R + power_L
```

```
energy_R = cumulative_trapezoid(power_R, t, initial=0)
```

```
energy_L = cumulative_trapezoid(power_L, t, initial=0)
```

```

energy_bil = cumulative_trapezoid(power_bil, t, initial=0)

print("=" * 60)

print("BILATERAL ENERGY SUMMARY (over full 4 s window)")

print("=" * 60)

print(f" Right leg total energy   : {energy_R[-1]:.3f} J")
print(f" Left leg total energy    : {energy_L[-1]:.3f} J")
print(f" Bilateral total energy    : {energy_bil[-1]:.3f} J")
print(f" Ratio bilateral / right    : {energy_bil[-1] / energy_R[-1]:.3f} (expected ~2.0)")

print()

#
=====

=====

# EXTENSION E: Baseline comparison and metabolic effort reduction estimate

#

# Rationale:

# The thesis proposal commits to "validation of reduction via simulations
# by at least 15%." To meet this, a baseline (unassisted) condition must
# be compared against the assisted condition.

#

# Surrogate metric used: Cumulative Squared Joint Torque (CSJT), defined as
# the integral of  $\omega^2$  over time. This is a standard proxy for muscle
# effort in simulation studies when EMG or metabolic measurement is not
# available. It represents the mechanical work the biological joint must
# perform. When the exoskeleton injects assistive torque during swing, the

```

```

# net biological effort required is reduced.

#

# Baseline (unassisted): full angular velocity squared integral, the joint
# must supply all torque itself.

# Assisted: the exoskeleton contributes torque_cmd during swing, so the
# residual biological torque demand is reduced by that amount.

# Reduction (%) = (baseline_effort - assisted_effort) / baseline_effort * 100

#
=====

=====

# Biological joint torque demand proxy:

# Unassisted: joint must produce torque proportional to velocity at all times

# (peak_assist_torque scale used for dimensional consistency)

unassisted_torque_demand = (np.abs(angular_velocity_rad) / max_vel_rad) * peak_assist_torque

# Assisted: exoskeleton offloads torque_cmd during swing;

# residual biological demand is reduced accordingly

assisted_torque_demand = np.maximum(unassisted_torque_demand - torque_cmd, 0.0)

# clamp at zero: biological torque cannot go negative (exoskeleton doesn't resist)

# Cumulative Squared Joint Torque (effort proxy, J equivalent)

# squaring penalises high-torque instants more heavily, consistent with metabolic cost models

baseline_effort = np.trapezoid(unassisted_torque_demand ** 2, t)

assisted_effort = np.trapezoid(assisted_torque_demand ** 2, t)

```

```

effort_reduction_pct = (baseline_effort - assisted_effort) / baseline_effort * 100.0

# Per-stride breakdown

baseline_stride_efforts = []

assisted_stride_efforts = []

for k in range(len(stride_starts) - 1):

    i0 = stride_starts[k]

    i1 = stride_starts[k + 1]

    baseline_stride_efforts.append(

        np.trapezoid(unassisted_torque_demand[i0:i1] ** 2, t[i0:i1])

    )

    assisted_stride_efforts.append(

        np.trapezoid(assisted_torque_demand[i0:i1] ** 2, t[i0:i1])

    )

baseline_stride_efforts = np.array(baseline_stride_efforts)

assisted_stride_efforts = np.array(assisted_stride_efforts)

per_stride_reduction = ((baseline_stride_efforts - assisted_stride_efforts)

                        / baseline_stride_efforts * 100.0)

print("=" * 60)

print("EXTENSION E: BASELINE vs ASSISTED EFFORT COMPARISON")

print("=" * 60)

print(f" Surrogate metric      : Cumulative Squared Joint Torque (Nm^2·s)")

```

```

print(f" Baseline effort      : {baseline_effort:.4f} Nm^2·s")

print(f" Assisted effort      : {assisted_effort:.4f} Nm^2·s")

print(f" Overall effort reduction : {effort_reduction_pct:.1f} %")

print()

print(f" Per-stride reduction    : {per_stride_reduction.mean():.1f} ± "

      f"{per_stride_reduction.std():.1f} %")

print()

# Check against 15% proposal target

if effort_reduction_pct >= 15.0:

    print(f" >>> Proposal target (>=15%) MET: {effort_reduction_pct:.1f}% reduction
    achieved.")

else:

    print(f" >>> Proposal target (>=15%) NOT MET: only {effort_reduction_pct:.1f}%
    achieved.")

print()

#
=====

=====

# FIGURES

# Figure 1: 4-panel control simulation (original)

# Figure 2: Parameter sensitivity analysis

# Figure 3: Bilateral simulation

# Figure 4: Baseline vs assisted effort comparison (new)

```

```
#
```

```
=====
```

```
# --- Figure 1: Original 4-panel plot ---
```

```
fig1, axs = plt.subplots(4, 1, figsize=(12, 10), sharex=True)
```

```
fig1.suptitle(
```

```
    'Exoskeleton Hierarchical Control Simulation: DSP, FSM, and Energy Analysis',
```

```
    fontsize=14, fontweight='bold'
```

```
)
```

```
axs[0].plot(t, noisy_hip_angle, color='lightgray', label='Raw Noisy Sensor Data')
```

```
axs[0].plot(t, filtered_hip_angle, color='navy', linewidth=2, label='Filtered Kinematics')
```

```
axs[0].set_ylabel('Hip Angle (deg)', fontsize=11)
```

```
axs[0].legend(loc='upper right')
```

```
axs[0].grid(True, linestyle='--', alpha=0.6)
```

```
axs[1].plot(t, state, color='black', linewidth=2)
```

```
axs[1].set_ylabel('Gait Phase', fontsize=11)
```

```
axs[1].set_yticks([0, 1])
```

```
axs[1].set_yticklabels(['Stance (0)', 'Swing (1)'])
```

```
axs[1].fill_between(t, 0, 1, where=(state == 1), color='lightgreen', alpha=0.3,
```

```
                    label='Assistance Window')
```

```
axs[1].legend(loc='upper right')
```

```
axs[1].grid(True, linestyle='--', alpha=0.6)
```

```

axs[2].plot(t, torque_cmd, color='firebrick', linewidth=2, label='Commanded Torque')

axs[2].set_ylabel('Torque (Nm)', fontsize=11)

axs[2].legend(loc='upper right')

axs[2].grid(True, linestyle='--', alpha=0.6)


axs[3].plot(t, exoskeleton_power, color='darkorange', linewidth=1.5,

            label='Mech. Power (W)')

axs[3].plot(t, injected_energy, color='purple', linewidth=2, linestyle='--',

            label='Cumulative Energy (J)')

axs[3].set_xlabel('Time (s)', fontsize=12)

axs[3].set_ylabel('Energetics', fontsize=11)

axs[3].legend(loc='upper left')

axs[3].grid(True, linestyle='--', alpha=0.6)


plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR, 'Figure_1_control_simulation.png'),

            dpi=150, bbox_inches='tight')


# --- Figure 2: Sensitivity analysis ---

fig2, (ax_a, ax_b) = plt.subplots(1, 2, figsize=(13, 5))

fig2.suptitle('Parameter Sensitivity Analysis', fontsize=13, fontweight='bold')


ax_a.plot(torque_sweep, sweep_energy, 'o-', color='navy', linewidth=2,

          markersize=6, label='Mean energy / stride')

ax_a.axvline(x=5.0, color='firebrick', linestyle='--', linewidth=1.5,

```

```

        label='Design point (5.0 Nm)')

ax_a.set_xlabel('Peak Assistance Torque (Nm)', fontsize=11)

ax_a.set_ylabel('Mean Energy per Stride (J)', fontsize=11)

ax_a.set_title('(a) Torque Sensitivity', fontsize=11)

ax_a.legend()

ax_a.grid(True, linestyle='--', alpha=0.6)


ax_a2 = ax_a.twinx()

ax_a2.plot(torque_sweep, sweep_peak_pwr, 's--', color='darkorange', linewidth=1.5,
           markersize=5, label='Mean peak power')

ax_a2.set_ylabel('Mean Peak Power (W)', fontsize=11, color='darkorange')

ax_a2.tick_params(axis='y', labelcolor='darkorange')

ax_a2.legend(loc='lower right')


ax_b.plot(deadband_sweep, db_swing_frac, 'o-', color='green', linewidth=2,
           markersize=6, label='Swing fraction (%)')

ax_b.axvline(x=0.1, color='firebrick', linestyle='--', linewidth=1.5,
             label='Design point (0.1 rad/s)')

ax_b.set_xlabel('FSM Deadband Threshold (rad/s)', fontsize=11)

ax_b.set_ylabel('Swing Phase Fraction (%)', fontsize=11)

ax_b.set_title('(b) Deadband Sensitivity', fontsize=11)

ax_b.legend()

ax_b.grid(True, linestyle='--', alpha=0.6)


ax_b2 = ax_b.twinx()

```

```

ax_b2.plot(deadband_sweep, db_energy, 's--', color='purple', linewidth=1.5,
           markersize=5, label='Energy / stride (J)')

ax_b2.set_ylabel('Mean Energy per Stride (J)', fontsize=11, color='purple')

ax_b2.tick_params(axis='y', labelcolor='purple')

ax_b2.legend(loc='upper right')

plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR, 'Figure_2_sensitivity_analysis.png'),
           dpi=150, bbox_inches='tight')

# --- Figure 3: Bilateral simulation ---

fig3, axs3 = plt.subplots(3, 1, figsize=(12, 8), sharex=True)

fig3.suptitle('Bilateral Hip Exoskeleton Simulation', fontsize=13, fontweight='bold')

axs3[0].plot(t, filt_R, color='navy', linewidth=2, label='Right hip (filtered)')
axs3[0].plot(t, filt_L, color='steelblue', linewidth=2, linestyle='--',
            label='Left hip (filtered)')

axs3[0].set_ylabel('Hip Angle (deg)', fontsize=11)

axs3[0].legend(loc='upper right')

axs3[0].grid(True, linestyle='--', alpha=0.6)

axs3[1].plot(t, torque_R, color='firebrick', linewidth=2, label='Right torque')
axs3[1].plot(t, torque_L, color='salmon', linewidth=2, linestyle='--',
            label='Left torque')

axs3[1].set_ylabel('Torque (Nm)', fontsize=11)

```

```

axs3[1].legend(loc='upper right')

axs3[1].grid(True, linestyle='--', alpha=0.6)


axs3[2].plot(t, power_R, color='darkorange', linewidth=1.5, label='Right power (W)')
axs3[2].plot(t, power_L, color='gold', linewidth=1.5, linestyle='--',
              label='Left power (W)')
axs3[2].plot(t, power_bil, color='black', linewidth=2,
              label='Total bilateral power (W)')
axs3[2].set_xlabel('Time (s)', fontsize=12)
axs3[2].set_ylabel('Mechanical Power (W)', fontsize=11)
axs3[2].legend(loc='upper right')
axs3[2].grid(True, linestyle='--', alpha=0.6)


plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR, 'Figure_3_bilateral_simulation.png'),
            dpi=150, bbox_inches='tight')


# --- Figure 4: Baseline vs assisted effort comparison ---
fig4, (ax4a, ax4b) = plt.subplots(1, 2, figsize=(13, 5))
fig4.suptitle('Baseline vs. Assisted Joint Effort Comparison',
              fontsize=13, fontweight='bold')


# Panel A: torque demand time series
ax4a.plot(t, unassisted_torque_demand, color='steelblue', linewidth=1.5,
          label='Unassisted (baseline)')

```

```

ax4a.plot(t, assisted_torque_demand, color='firebrick', linewidth=1.5,
          linestyle='--', label='Assisted (exoskeleton)')

ax4a.fill_between(t, assisted_torque_demand, unassisted_torque_demand,
                  alpha=0.15, color='green', label='Effort reduction region')

ax4a.set_xlabel('Time (s)', fontsize=11)

ax4a.set_ylabel('Joint Torque Demand Proxy (Nm)', fontsize=11)

ax4a.set_title('(a) Torque Demand: Baseline vs. Assisted', fontsize=11)

ax4a.legend()

ax4a.grid(True, linestyle='--', alpha=0.6)

# Panel B: per-stride reduction bar chart

stride_labels = [f'Stride {k+1}' for k in range(len(per_stride_reduction))]

bars = ax4b.bar(stride_labels, per_stride_reduction, color='steelblue',
               edgecolor='navy', alpha=0.8)

ax4b.axhline(y=15.0, color='firebrick', linestyle='--', linewidth=1.5,
             label='Proposal target (15%)')

ax4b.axhline(y=effort_reduction_pct, color='green', linestyle='-', linewidth=1.5,
             label=f'Overall reduction ( {effort_reduction_pct:.1f} %)')

ax4b.set_ylabel('Effort Reduction (%)', fontsize=11)

ax4b.set_title('(b) Per-Stride Effort Reduction', fontsize=11)

ax4b.set_ylim(0, max(per_stride_reduction.max(), 20) * 1.2)

ax4b.legend()

ax4b.grid(True, linestyle='--', alpha=0.6, axis='y')

# Annotate bars

```

```
for bar, val in zip(bars, per_stride_reduction):

    ax4b.text(bar.get_x() + bar.get_width() / 2.0, bar.get_height() + 0.3,

              f'{val:.1f}%', ha='center', va='bottom', fontsize=10, fontweight='bold')


plt.tight_layout()

plt.savefig(os.path.join(OUTPUT_DIR, 'Figure_4_baseline_comparison.png'),

            dpi=150, bbox_inches='tight')


plt.show()

print("All figures saved to:", OUTPUT_DIR)
```

Appendix B

```

import time

import odrive

from odrive.enums import (

    AXIS_STATE_IDLE,

    AXIS_STATE_FULL_CALIBRATION_SEQUENCE,

)


# =====

# USER SETTINGS

# =====

SERIAL_NUMBER = "356935703334"


# Motor / encoder

POLE_PAIRS = 14

ENCODER_CPR = 2000


# Current limits

CALIBRATION_CURRENT = 1.0    # A

CURRENT_LIMIT = 5.0         # A


# Brake resistor / DC bus

ENABLE_BRAKE_RESISTOR = True

BRAKE_RESISTANCE = 2.0      # ohm

MAX_REGEN_CURRENT = 0.0     # A

```

```
DC_MAX_NEGATIVE_CURRENT = -0.1 # A
```

```
def main():
```

```
    print("Connecting to ODrive...")
```

```
    odrv = odrive.find_any(serial_number=SERIAL_NUMBER, timeout=10)
```

```
    print("Connected.")
```

```
    odrv.clear_errors()
```

```
    # ----- ODrive-level DC / brake settings -----
```

```
    odrv.config.enable_brake_resistor = ENABLE_BRAKE_RESISTOR
```

```
    odrv.config.brake_resistance = BRAKE_RESISTANCE
```

```
    odrv.config.max_regen_current = MAX_REGEN_CURRENT
```

```
    odrv.config.dc_max_negative_current = DC_MAX_NEGATIVE_CURRENT
```

```
    odrv.axis0.encoder.config.cpr = 4000
```

```
    odrv.axis1.encoder.config.cpr = 2000
```

```
    for axis_name, ax in [("axis0", odrv.axis0), ("axis1", odrv.axis1)]:
```

```
        print(f"\nApplying configuration to {axis_name}...")
```

```
        ax.requested_state = AXIS_STATE_IDLE
```

```
        time.sleep(0.1)
```

```
    # ----- Axis / motor / encoder -----
```

```
    ax.motor.config.pole_pairs = POLE_PAIRS
```

```
    ax.motor.config.calibration_current = CALIBRATION_CURRENT
```

```

ax.motor.config.current_lim = CURRENT_LIMIT

ax.motor.config.phase_resistance = 2.2

ax.motor.config.phase_inductance = 1.33e-3

ax.controller.config.vel_limit = 12

ax.motor.config.torque_constant = 0.127

ax.motor.config.torque_lim = 0.05


print(f'Starting FULL CALIBRATION SEQUENCE for {axis_name}...')

ax.requested_state = AXIS_STATE_FULL_CALIBRATION_SEQUENCE


t0 = time.time()

start_timeout = 5.0

while ax.current_state == AXIS_STATE_IDLE and (time.time() - t0) < start_timeout:

    time.sleep(0.05)


if ax.current_state == AXIS_STATE_IDLE:

    print(f'{axis_name}: Calibration did not start.')

    print(f'{axis_name}.axis.error = {ax.error}')

    print(f'{axis_name}.motor.error = {ax.motor.error}')

    print(f'{axis_name}.encoder.error = {ax.encoder.error}')

    continue


print(f'{axis_name}: Calibration started...')


t1 = time.time()

```

```

finish_timeout = 60.0

while ax.current_state != AXIS_STATE_IDLE and (time.time() - t1) < finish_timeout:

    time.sleep(0.1)

print(f"\n{axis_name}: Calibration finished.")

print(f"{axis_name}.axis.error = {ax.error}")

print(f"{axis_name}.motor.error = {ax.motor.error}")

print(f"{axis_name}.encoder.error = {ax.encoder.error}")

if ax.current_state != AXIS_STATE_IDLE:

    print(f"{axis_name}: Result: TIMEOUT (axis did not return to IDLE)")

elif ax.error == 0 and ax.motor.error == 0 and ax.encoder.error == 0:

    print(f"{axis_name}: Result: FULL CALIBRATION SUCCESS")

else:

    print(f"{axis_name}: Result: FULL CALIBRATION FAILED")

if __name__ == "__main__":

    main()

```

Appendix C

```

import time

import odrive

import numpy as np

import matplotlib.pyplot as plt

from odrive.enums import (

    AXIS_STATE_IDLE,

    AXIS_STATE_CLOSED_LOOP_CONTROL,

    CONTROL_MODE_VELOCITY_CONTROL,

    INPUT_MODE_PASSTHROUGH,

)

# =====

# USER SETTINGS

# =====

SERIAL_NUMBER = "356935703334"


# Motor / encoder

POLE_PAIRS = 14

ENCODER_CPR = 2000


# Current limits

CALIBRATION_CURRENT = 1.0    # A

CURRENT_LIMIT = 5.0          # A

```

Brake resistor / DC bus

ENABLE_BRAKE_RESISTOR = True

BRAKE_RESISTANCE = 2.0 # ohm

MAX_REGEN_CURRENT = 0.0 # A

DC_MAX_NEGATIVE_CURRENT = -0.1 # A

Motion / control

GEAR_RATIO = 10.0 # motor turns / output turn

AMP_DEG = 40.0 # output-shaft amplitude

F_REF = 1.0 # Hz

DT = 0.01 # s

RUN_TIME = 30.0 # s

VEL_LIMIT_MOTOR = 12.0 # turns/s (motor shaft)

KP_POS = 0.08 # output-deg -> motor-turn/s correction

Optional save

SAVE_CONFIG = False

REBOOT_AFTER_SAVE = False

def main():

print("Connecting to ODrive...")

odrv = odrive.find_any(serial_number=SERIAL_NUMBER, timeout=10)

print("Connected.")

odrv.clear_errors()

```

axes = {

    "axis0": odrv.axis0,

    "axis1": odrv.axis1,

}

print("Applying configuration...")

# ----- ODrive-level DC / brake settings -----

odrv.config.enable_brake_resistor = ENABLE_BRAKE_RESISTOR

odrv.config.brake_resistance = BRAKE_RESISTANCE

odrv.config.max_regen_current = MAX_REGEN_CURRENT

odrv.config.dc_max_negative_current = DC_MAX_NEGATIVE_CURRENT

odrv.axis0.encoder.config.cpr = 4000

odrv.axis1.encoder.config.cpr = 2000

# ----- Axis / motor / encoder -----

for name, ax in axes.items():

    ax.requested_state = AXIS_STATE_IDLE

    time.sleep(0.1)

    ax.motor.config.pole_pairs = POLE_PAIRS

    ax.motor.config.calibration_current = CALIBRATION_CURRENT

    ax.motor.config.current_lim = CURRENT_LIMIT

    ax.controller.config.vel_limit = 12

```

```

ax.controller.config.vel_gain = 0.08

ax.controller.config.vel_integrator_gain = 0.11

ax.motor.config.torque_constant = 0.127

ax.motor.config.torque_lim = 0.03


print(f"{name} configured.")


print("Configuration applied.\n")


if SAVE_CONFIG:

    print("\nSaving configuration...")

    odrv.save_configuration()

    print("Save command sent.")

    if REBOOT_AFTER_SAVE:

        print("Waiting for reboot...")

        time.sleep(5)


# ----- Velocity control mode -----

for name, ax in axes.items():

    ax.controller.config.control_mode = CONTROL_MODE_VELOCITY_CONTROL

    ax.controller.config.input_mode = INPUT_MODE_PASSTHROUGH

    ax.requested_state = AXIS_STATE_CLOSED_LOOP_CONTROL


# Verify both axes really entered closed loop

time.sleep(0.3)

```

```

for name, ax in axes.items():

    t0 = time.time()

    while ax.current_state != AXIS_STATE_CLOSED_LOOP_CONTROL and (time.time() -
t0) < 3.0:

        time.sleep(0.05)

    if ax.current_state != AXIS_STATE_CLOSED_LOOP_CONTROL:

        print(f"\n{name} failed to enter CLOSED_LOOP_CONTROL")

        print(f"{name}.current_state = {ax.current_state}")

        print(f"{name}.axis.error   = {ax.error}")

        print(f"{name}.motor.error  = {ax.motor.error}")

        print(f"{name}.encoder.error = {ax.encoder.error}")

        return

# Use current output position as sinusoid center for each axis

q_center_deg = {

    "axis0": axes["axis0"].encoder.pos_estimate * 360.0 / GEAR_RATIO,

    "axis1": axes["axis1"].encoder.pos_estimate * 360.0 / GEAR_RATIO,

}

print("\nClosed loop enabled on both axes.")

print(f"axis0 center position = {q_center_deg['axis0']:.2f} deg")

print(f"axis1 center position = {q_center_deg['axis1']:.2f} deg")

print(f"Tracking same sinusoid on both axes: amplitude = {AMP_DEG} deg, frequency =
{F_REF} Hz")

```

```

# ----- Live plot -----

plt.ion()

fig, ((ax0_pos, ax1_pos), (ax0_vel, ax1_vel)) = plt.subplots(2, 2, figsize=(14, 8), sharex=True)

l_q0, = ax0_pos.plot([], [], label="axis0 q actual (deg)")
l_q0ref, = ax0_pos.plot([], [], "--", label="axis0 q ref (deg)")
ax0_pos.set_ylabel("Position (deg)")
ax0_pos.set_title("axis0 position")
ax0_pos.legend(loc="upper left")
ax0_pos.grid(True)

l_q1, = ax1_pos.plot([], [], label="axis1 q actual (deg)")
l_q1ref, = ax1_pos.plot([], [], "--", label="axis1 q ref (deg)")
ax1_pos.set_ylabel("Position (deg)")
ax1_pos.set_title("axis1 position")
ax1_pos.legend(loc="upper left")
ax1_pos.grid(True)

l_dq0, = ax0_vel.plot([], [], label="axis0 dq actual (deg/s)")
l_dq0ref, = ax0_vel.plot([], [], "--", label="axis0 dq ref (deg/s)")
ax0_vel.set_ylabel("Velocity (deg/s)")
ax0_vel.set_xlabel("Time (s)")
ax0_vel.set_title("axis0 velocity")
ax0_vel.legend(loc="upper left")
ax0_vel.grid(True)

```

```

l_dq1, = ax1_vel.plot([], [], label="axis1 dq actual (deg/s)")
l_dq1ref, = ax1_vel.plot([], [], "--", label="axis1 dq ref (deg/s)")
ax1_vel.set_ylabel("Velocity (deg/s)")
ax1_vel.set_xlabel("Time (s)")
ax1_vel.set_title("axis1 velocity")
ax1_vel.legend(loc="upper left")
ax1_vel.grid(True)

t_hist = []
q0_hist, q0ref_hist, dq0_hist, dq0ref_hist = [], [], [], []
q1_hist, q1ref_hist, dq1_hist, dq1ref_hist = [], [], [], []

MAX_SAMPLES = 4000
PLOT_DT = 0.05
t_plot_last = time.time()

omega = 2.0 * np.pi * F_REF
t0 = time.time()

print("\nRunning ... close the plot window or wait for RUN_TIME to stop.\n")

try:
    while True:
        t = time.time() - t0

```

```

if t >= RUN_TIME:

    break

if not plt.fignum_exists(fig.number):

    break


# Output-side sinusoidal reference for both axes

q0_ref_deg = q_center_deg["axis0"] + AMP_DEG * np.sin(omega * t)
q1_ref_deg = q_center_deg["axis1"] + AMP_DEG * np.sin(omega * t)
dq_ref_deg_s = AMP_DEG * omega * np.cos(omega * t)


# Read actual output position / velocity

q0_deg = axes["axis0"].encoder.pos_estimate * 360.0 / GEAR_RATIO
dq0_deg_s = axes["axis0"].encoder.vel_estimate * 360.0 / GEAR_RATIO
q1_deg = axes["axis1"].encoder.pos_estimate * 360.0 / GEAR_RATIO
dq1_deg_s = axes["axis1"].encoder.vel_estimate * 360.0 / GEAR_RATIO


# Velocity command conversion for axis0

pos0_err_deg = q0_ref_deg - q0_deg
vel0_ff_motor = dq_ref_deg_s * GEAR_RATIO / 360.0
vel0_fb_motor = KP_POS * pos0_err_deg * GEAR_RATIO / 360.0
vel0_cmd_motor = vel0_ff_motor + vel0_fb_motor

vel0_cmd_motor = float(np.clip(vel0_cmd_motor, -VEL_LIMIT_MOTOR,
VEL_LIMIT_MOTOR))


# Velocity command conversion for axis1

```

```

pos1_err_deg = q1_ref_deg - q1_deg

vel1_ff_motor = dq_ref_deg_s * GEAR_RATIO / 360.0

vel1_fb_motor = KP_POS * pos1_err_deg * GEAR_RATIO / 360.0

vel1_cmd_motor = vel1_ff_motor + vel1_fb_motor

vel1_cmd_motor = float(np.clip(vel1_cmd_motor, -VEL_LIMIT_MOTOR,
VEL_LIMIT_MOTOR))

```

```

axes["axis0"].controller.input_vel = vel0_cmd_motor

axes["axis1"].controller.input_vel = vel1_cmd_motor

```

```

# Log

```

```

t_hist.append(t)

```

```

q0_hist.append(q0_deg)

q0ref_hist.append(q0_ref_deg)

dq0_hist.append(dq0_deg_s)

dq0ref_hist.append(dq_ref_deg_s)

```

```

q1_hist.append(q1_deg)

q1ref_hist.append(q1_ref_deg)

dq1_hist.append(dq1_deg_s)

dq1ref_hist.append(dq_ref_deg_s)

```

```

if len(t_hist) > MAX_SAMPLES:

```

```

    t_hist.pop(0)

```

```
q0_hist.pop(0)
```

```
q0ref_hist.pop(0)
```

```
dq0_hist.pop(0)
```

```
dq0ref_hist.pop(0)
```

```
q1_hist.pop(0)
```

```
q1ref_hist.pop(0)
```

```
dq1_hist.pop(0)
```

```
dq1ref_hist.pop(0)
```

```
# Plot refresh
```

```
now = time.time()
```

```
if now - t_plot_last >= PLOT_DT:
```

```
    t_plot_last = now
```

```
tv = np.array(t_hist)
```

```
q0v = np.array(q0_hist)
```

```
q0rv = np.array(q0ref_hist)
```

```
dq0v = np.array(dq0_hist)
```

```
dq0rv = np.array(dq0ref_hist)
```

```
q1v = np.array(q1_hist)
```

```
q1rv = np.array(q1ref_hist)
```

```

dq1v = np.array(dq1_hist)

dq1rv = np.array(dq1ref_hist)


l_q0.set_data(tv, q0v)
l_q0ref.set_data(tv, q0rv)
ax0_pos.relim()
ax0_pos.autoscale_view()


l_q1.set_data(tv, q1v)
l_q1ref.set_data(tv, q1rv)
ax1_pos.relim()
ax1_pos.autoscale_view()


l_dq0.set_data(tv, dq0v)
l_dq0ref.set_data(tv, dq0rv)
ax0_vel.relim()
ax0_vel.autoscale_view()


l_dq1.set_data(tv, dq1v)
l_dq1ref.set_data(tv, dq1rv)
ax1_vel.relim()
ax1_vel.autoscale_view()


fig.canvas.draw_idle()
fig.canvas.flush_events()

```

```
time.sleep(DT)

finally:

    axes["axis0"].controller.input_vel = 0.0

    axes["axis1"].controller.input_vel = 0.0

    time.sleep(0.05)

    axes["axis0"].requested_state = AXIS_STATE_IDLE

    axes["axis1"].requested_state = AXIS_STATE_IDLE

    print("Motors set to IDLE.")

    plt.ioff()

    plt.show()

if __name__ == "__main__":

    main()
```