University of Toronto
Faculty of Applied Science and Engineering
APS490 - Capstone
Final Report

Project #	TRI 5	Date	March 24th, 2024

Project Title	Actuator Modelling for Simulation of Energy-Efficient Human-Robot Walking
Client	Professor Brokoslaw Laschowski
Supervisor	Professor Hugh Liu
Team Lead	Ugo Jalleau
Client Contact Person	Stephen Yang, Amreen Imrit
Project Manager	Yvonne Yang
Research Lead	Koby Lee
Prepared By (Names and Student #s of Team Members)	Stephen Yang (1005917643) Yvonne Yang (1006084381) Koby Lee (1005831209) Ugo Jalleau (1006209345) Amreen Imrit (1005902971)

Table of Contents

Executive Summary

This project focuses on advancing robotic prostheses by characterizing the energy efficiency of energy-regenerative brushless DC motors. It leverages a multi-disciplinary engineering approach, using both experimental data with a dynamometer setup and mathematical modeling for theoretical prediction. The hope is for the results to inform future controller designs to extend battery life and improve device usability.

Throughout the design process, the team faced many mechanical, electrical, and software challenges with the dynamometer setup. Mechanical challenges, such as gear shearing and motor coil damage, were carefully addressed through material enhancements and redesigned components, ensuring durability in operation. Additionally, electrical issues, including torn connectors and voltage irregularities, prompted the implementation of temporary fixes and ongoing monitoring protocols to maintain system integrity. The software architecture involves a C++ codebase for automating the testing sequence and Python scripts for post-processing and real-time GUI display, facilitating data analysis and visualization. The code was reorganized and refactored for improved performance, readability, and maintainability. Most importantly, the C++ code automates a testing procedure that goes through various operating points for generating the energy efficiency graph, and the functionality to interface with two controllers in different control modes was developed. Efficiency calculations are based on real-time data obtained during the "hold" stages of each test case. A specific process is followed to obtain relevant data points, and efficiency is calculated based on equations from relevant literature. The project aims to integrate real-time data from the dynamometer into the graphical user interface (GUI) for more realistic dynamic graphical output during presentations.

The mathematical modeling framework offers a versatile approach to assessing actuator efficiency across diverse input types, accommodating both discrete and continuous data alongside time-dependent or independent variables. Through consistent principles governing DC motor model, efficiency calculations and strategic mitigation of discontinuities and outliers, the simulation delivers insights into actuator performance and energy regeneration potential. Comparative analyses to [1] underscore its utility in predicting outcomes and informing design decisions, yet ongoing refinement and validation remain imperative to fortify model accuracy for future applications and research endeavors.

Looking ahead, future work will focus on completing mathematical modeling for controller design, implementing mechanical and electrical enhancements, and refining software systems to improve tracking accuracy and stability. Challenges in the software system, such as torque controller issues and GUI script alignment, necessitate further attention and iterative refinement. Unit testing for controllers and parameter refinement are recommended for system robustness and reliability, ensuring seamless integration into real-world applications.

1.0 Introduction

1.1 Problem Statement

In robotics, the task of increasing operation time in spite of limited battery capacity is a relentless and persistent challenge. Research in power and actuator systems shows that bidirectional actuators (i.e. brushless DC motors) may be back-driven to recharge an electromechanical system. As people move, their joints are subject to negative work from the environment, such as when descending stairs or stopping after a jog. In wearable robotics, this mechanical energy can be transformed into electrical energy back into the system [1]. This is analogous to regenerative braking in cars, where kinetic energy is harnessed during braking, converted into electrical energy, and stored in the vehicle's battery for later use. The main challenge in energy regeneration is the development and tuning of a control system that identifies the operating energy state and sends out the appropriate commands; said system requires an accurate mathematical model that can predict the behavior of the exoskeleton over and past its range of operation. The scope of this project is to characterize the energy regeneration capabilities of a single actuator used in the exoskeleton, with the ultimate goal of creating a predictive model of the actuator's energy efficiency in all quadrants of operation (motoring and braking); details about these operation states are presented in Section 1.2. The method involves combining efficiency data from a dynamometer testing setup with results from a physics-based simulation. This study focuses on the open-source 3D printed actuators designed in [2]. Appendix A provides a visual for the chronological problem definition.

1.2 Existing Work and Contributions

To choose the various parameters of a testing rig, it is crucial to understand the actuator's torque-speed relationship. As seen in Figure 1, the actuator operates in four distinct quadrants: forward motoring, forward braking, reverse motoring, and reverse braking. Provided the motor drive allows bi-directional power flow, energy can flow from the motor to the load (quadrants I and III), or from the load to the motor (quadrants II and IV) [3]. During motoring operation, the efficiency is measured as the ratio between the mechanical power output and electrical power input –vice versa during braking. There are various ways to model these efficiencies depending on the motor-and-gearbox setup and the underlying power dynamics [3, 4]. In human gait, there are many periods of negative work (braking) with potential for energy regeneration; characterizing the actuator for power efficiency in these periods is crucial to understanding how much energy can flow back to the battery.

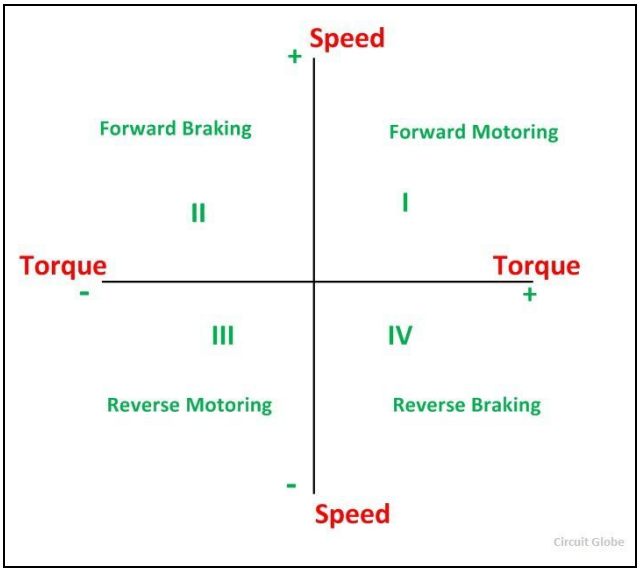


Figure 1: Operating Quadrants of an Actuator

Urs [2] presents open-source designs for two quasi-direct-drive actuators suitable for robots weighing 8–15 kg. These actuators are constructed using off-the-shelf and 3D-printed components, costing less than $200 each. The paper details the mechanical, electrical, and thermal properties of the actuators, demonstrating their durability with only a 2% efficiency reduction and 26 milliradians backlash growth after 420k strides. The performance of these actuators is comparable to traditional machined actuators, making them a cost-effective and customizable solution for high-speed legged robots. The Urs design has only been evaluated by himself and his lab, and has yet to be used in other designs. Additionally, there are disparities between the torque and speed achievable by the actuator and those typical of human operation. Thus the team endeavored to both test the usability of this 3D printed actuator design, as well as develop a simulation model to assess its performance when applied to the human gait cycle.

Laschowski [1] explores the potential of backdriveable actuators with energy regeneration to enhance the efficiency and extend the battery life of robotic lower-limb exoskeletons. By generating electrical energy from converted mechanical energy gained during the phases of negative work in the walking cycle, the study simulates energy regeneration across various walking speeds and slopes. The findings suggest that energy regeneration at slower walking speeds on declines could significantly increase the operating time of exoskeletons; it can improve locomotor efficiency by up to 99% in terms of the total number of steps. The research extends beyond previous studies that focused only on level-ground walking, offering insights into daily locomotor activities and real-world mobility. One of the limitations of the study was that the motor could not regenerate energy. Thus, the efficiency in the forward direction of motion was assumed to be the same as in the backwards direction. The model proposed by the team plans to test this assumption.

Verstraten [4] examines the energy efficiency of geared DC motors in dynamic applications, comparing theoretical models with experimental data. The paper highlights the inconsistencies in how energy consumption is calculated and the often neglected speed/ load dependent losses. It presents a case study of an 80W geared DC motor applying a sinusoidal trajectory to a pendulum, formulating general recommendations for modeling energy losses. The findings demonstrate that commonly used models can

predict energy consumption with less than 10% error when compared to experimental measurements, emphasizing the importance of accurate actuator models for optimizing power consumption. The equations used by Verstraten are generalized for any motor so the goal was thus to apply these fundamental equations to the Urs actuator. The main goal was to recreate Verstraten's heat efficiency map with the motor torque and speed values from Urs's actuator in order to understand how applicable the actuator design would be for human walking conditions.

1.3 System Diagram

This section highlights the various aspects of the work done by the team, and will reference the flowchart in Figure 2. As mentioned, the goal of this project is to develop a predictive model of the actuator's energy efficiency across all quadrants of operation. Since the focus is on regenerating energy during human gait, the inputs of the model are torque and speed while the outputs are efficiency maps of said torque and speed. In addition, the efficiency values are plotted against the stride time to highlight periods of negative work, where energy is available for regeneration. The modeling process was divided into two main sections: the physical model and the virtual simulation. The physical model refers to the benchtop dynamometer setup, which is a testing device with two actuators connected to each other through a torque sensor to measure torque-speed profiles; the details of the electromechanical system are highlighted in Section 2.2 and 2.3. The other part of the project focuses on simulating the theoretical physical behavior of the actuator, and was derived mathematically using a pendulum set-up. Details about the system and model are outlined in Section 3.

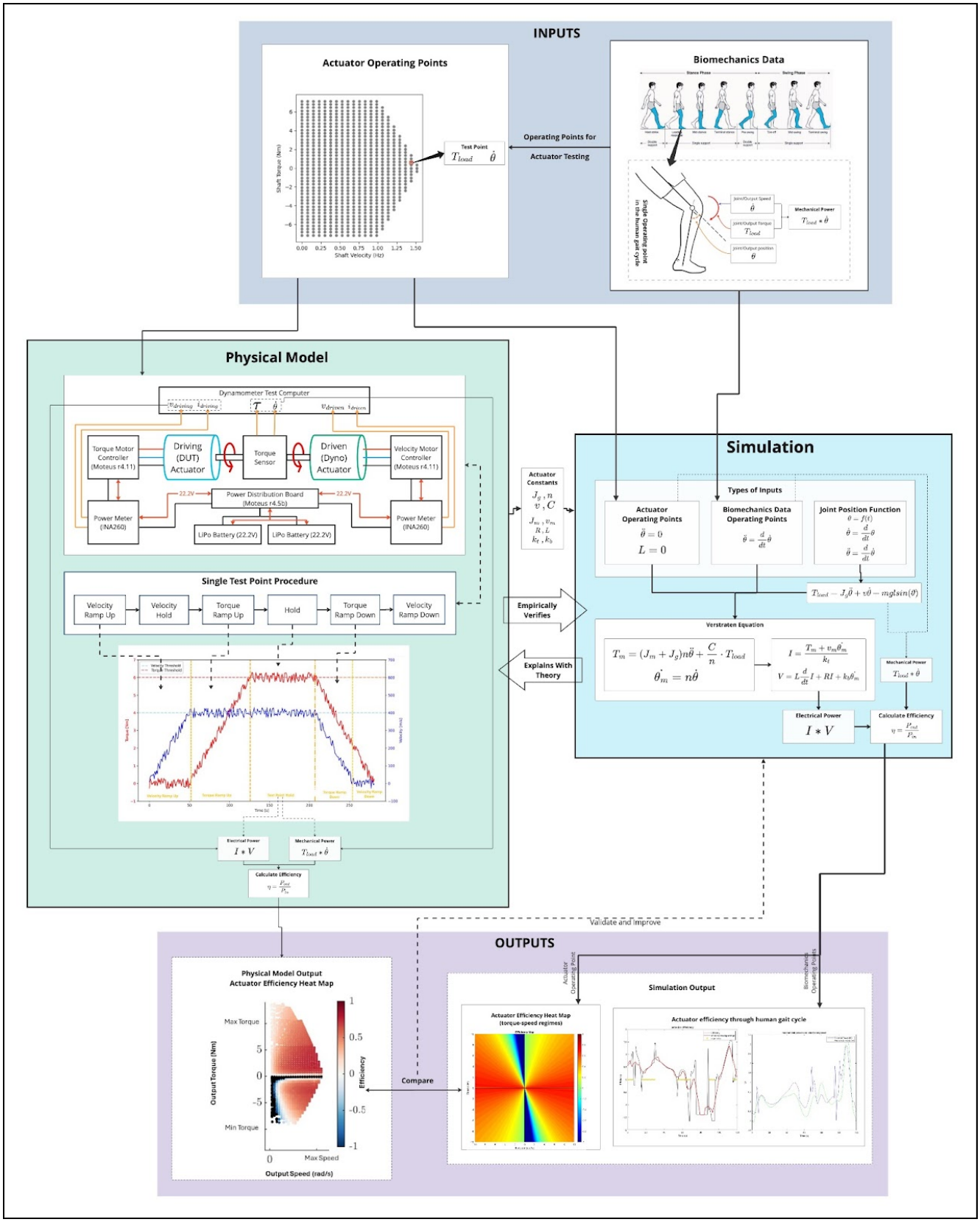


Figure 2: Actuator Testing and Modeling Workflow

To better understand the implications of the benchtop mechanism and the simulation, it is important to understand the system inputs and desired outputs. In [1], joint powers were derived from open-source biomechanics data using rigid-body inverse dynamics; these values were then integrated over time to yield profiles of mechanical energy absorption and dissipation during human locomotion. This dataset provides torque-speed pairs that represent discrete knee states during gait. These operating points can be fed to the benchtop mechanism to yield an efficiency map of the actuator (electrical power is measured at the motor terminals). On the other hand, the human gait can be represented as a continuous function of knee angle or speed. The simulation allows for mechanical power to be derived from first principles using ordinary differential equations, while the electrical power is calculated from a simple DC motor model. Here, the efficiency profile is outputted as a heat map and a time-based function. Through system identification, an accurate mathematical model can be derived by combining the simulation results to the dynamometer data. This project's next steps are further discussed in Section 4.0.

The components of the benchtop dynamometer are highlighted under "Physical Model" in Figure 2. The experimental setup uses two actuators connected at the shaft through a torque sensor, which logs values for velocity and torque. The motor controllers and power meters provide velocity and current values for the driven and driving motor. In this case, the driven motor acts as the load against the driving motor; the data is collected from the driving motor, which simulates the knee joint. The driven motor is operated in closed-loop velocity control while the driving motor is operated in open loop torque control. To collect data, the following test procedure was developed: velocity of the driven motor is ramped up to the desired value, torque of the driving motor is ramped up to the desired value, both are held for a set time (note, this is referred to as a torque-speed operating point), then torque and velocity are ramped down in that order. While conducting the experiments, the team encountered hardware issues, namely gear shredding, actuator housing stripping, and motor coils snapping. In addition, the microcontrollers suffered damage that significantly slowed down the software development and testing process.

The simulation was developed for use in conjunction with the benchtop dynamometer to create a realistic model of the actuator. Once validated, the combined mathematical model can be used to extrapolate efficiency values beyond the operating range of the motors. To achieve this, the actuator was modeled as a motor-gearbox-shaft system subject to pendulum loading. This setup was chosen for its applicability to knee joint dynamics, where the knee is isolated as a member with a torque applied at the extremity (foot); the governing differential equations are highlighted in Figure 2 under "Simulation". The simulation was generalized to three types of inputs: actuator operating points, biomechanics data, and joint position functions. Following the DC motor model, the efficiency was calculated through all four quadrants of operation. Similar to [1], it can be seen that regeneration decreases with incline. However, compared to [1], the total number of steps from the same battery capacity is significantly lower while the efficiency values are much higher. Likely, differences in joint modeling (degree of freedom) and discrepancies in data are responsible for the mismatch. Because of the many issues encountered with the benchtop, the team was unable to perform system identification on the actuator to optimize the mathematical model.

2.0 Benchtop

The benchtop is a dynamometer setup, utilizing two motors and various sensors to enable testing of an actuator. With one actuator (dynamo actuator, or driven actuator) acting as a variable load, and one actuator (test actuator, or driving actuator) acting as a device under test, the benchtop allows isolated testing of the driving actuator without concerning other components of the robotic prosthesis. The goal of the benchtop testing is to map out the actuator efficiency over torque and velocity regimes in the 4 quadrants of motor operation.

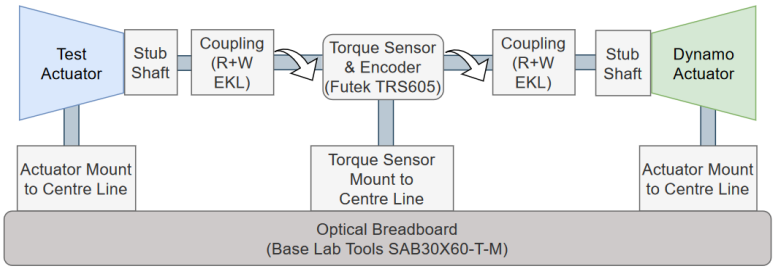


Figure 3: Mechanical Design of Benchtop [6]

2.1 Experiment Setup

To obtain the necessary values to calculate the actuator efficiency over a range of velocity, the benchtop is set up so that the driven actuator is operated in closed loop velocity control while the driving actuator is operated in open loop torque control. The goal of the experiment is to measure the final torque reached by the test, or driving, actuator. To extract an efficiency value for a given torque velocity operating point, the instantaneous current and voltage must also be recorded for each actuator. Then, the efficiency for a given torque-velocity operating point can be expressed as:

$$\eta = \frac{P_{out}}{P_{in}} = \frac{\tau \ \dot{\theta}}{iv} \tag{1}$$

Where $\dot{\theta}$ is the velocity of the shaft, τ is the torque reached by the driving actuator, i is the driving actuator current, and v is the driving actuator voltage. Figure 4 below [6] shows the test procedure to obtain the desired values from the actuators and sensors.

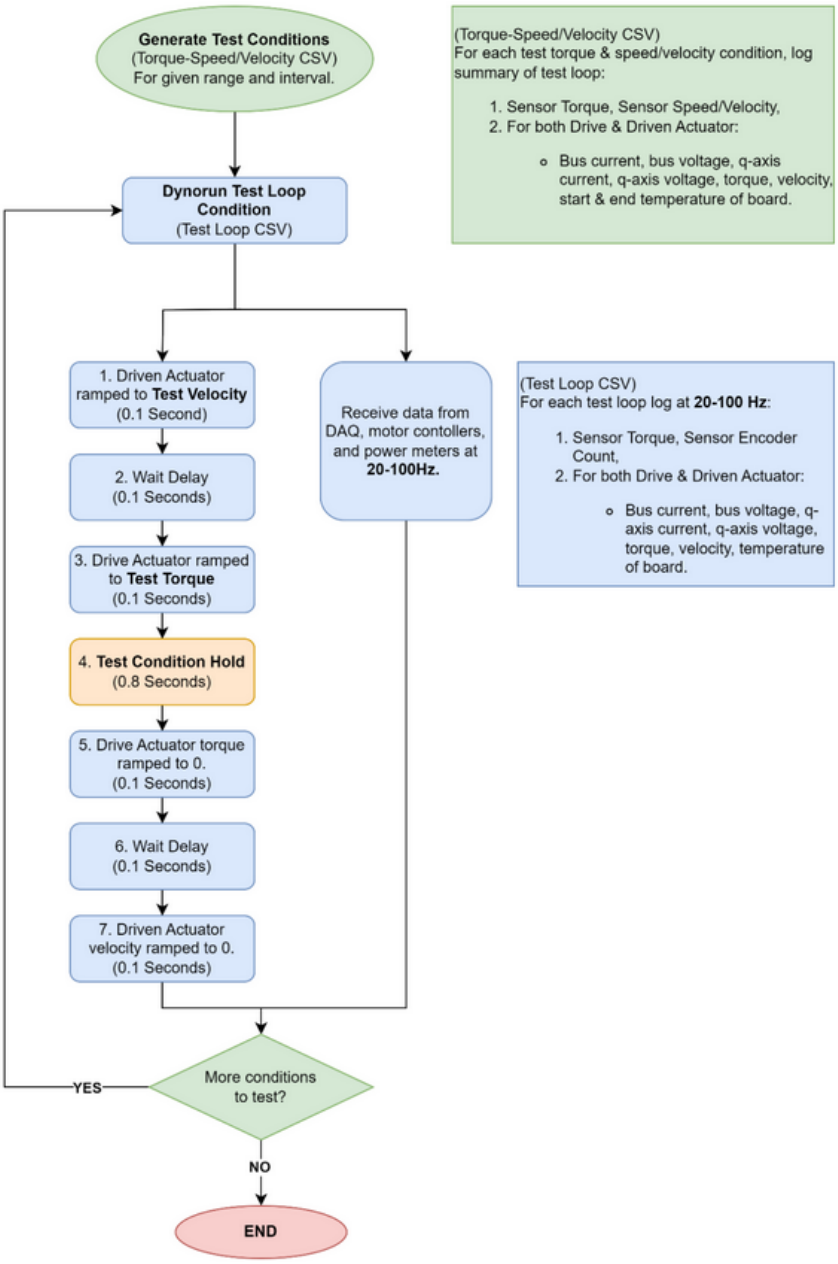


Figure 4: Flowchart of Testing Procedure [6]

The actuators are not operated under closed-loop torque control, as this would require constant acceleration exceeding the preset velocity limit, which may damage the actuators. In this setup, the velocity input represents an operating point (speed of human joint at a specific time in the human gait cycle) while the torque represents an external reference signal (such as a torque supplied by an external prosthetic motor). The driven actuator's velocity is the independent variable while the final torque of the driving actuator is the dependent variable. In this scenario, we do not expect the driving actuator to reach the desired torque value since it is open loop controlled, and thus does not receive any feedback sensor to adjust until the desired torque is reached.

2.1.1 Data setup for experiment

The torque limit of the actuator is ~10 Nm and the velocity limit is 10 rad/s (1.59 Hz) [2]. Beyond these limits the actuator will be operating past its designed operating point and may experience failure faster than expected. Using the efficiency map from [2], and taking into consideration the saturation of the motor, the team decided on the following for the test point distribution. Since the benchtop records speed as Hz instead of rad/s, the following rules assume velocity is in Hz.

1. The maximum velocity tested will be 1.59 Hz, with a spacing of 0.07 Hz between each velocity point
2. The maximum torque tested will be 10 Nm, with a spacing of 0.33 Nm between each torque point
3. No torque-velocity point shall be above the line

$$Torque = -12.66 * velocity + 20.128 \quad (2)$$

4. The point 0,0 does not need to be tested, but points (0, torque) and (velocity, 0), where torque and velocity is non-zero will be tested.

Following these rules, the torque-velocity points will be as presented in Figure 5. However, it should be noted that the spacing of the velocity and torque can change depending on the precision and noisiness of the data.

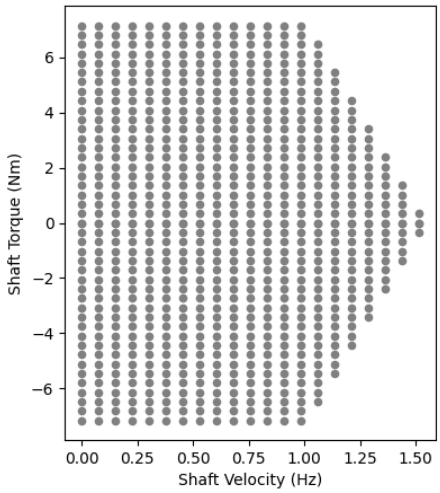


Figure 5: Velocity-Torque Test Points for Benchtop Testing

2.2 Mechanical System

The design for the actuators tested on the benchtop was sourced from Urs et al [2]. All custom components in the design are manufactured using 3D printing. The majority of these parts are produced through fused-deposition (FDM) printing, utilizing polylactic acid (PLA) plastic and printed on a Prusa MK3S+ printer. Despite FDM-PLA parts lacking mechanical strength compared to molded plastic or metals, designs that minimize stress concentrations and optimize print orientation have demonstrated sufficient strength for the intended application. To address concerns of elevated temperatures near the motor exceeding the glass transition temperature of PLA, certain parts are manufactured via stereolithography (SLA) using High Temp Resin with a Formlabs Form 2 printer.

The actuator is divided into two subassemblies. The Input Assembly includes the RI50 motor, a cooling solution, and an open-source motor driver (moteus r4.5). All these components are mounted onto a SLA-HT motor housing. The RI50 motor rotor is connected to the rotor-stator gear (RSG), which is reinforced with a steel dowel pin and houses a magnet for interacting with an encoder. This component bridges the input and transmission assembly. In the case of the 7.5:1 planetary transmission, the planet carrier serves as the output. For the bilateral drive transmission, the secondary ring gear acts as the output. All gears employ a 30-degree pressure angle involute profile to enhance strength. The main housings align with each other using an annular boss and are joined axially with eight screws.

Although the mechanical system was already assembled, the team experienced some issues when working with the actuators. The failure modes of these actuators and our corresponding solutions will be discussed in the following sections.

2.2.1 Sheared Gear

On February 5th, while operating the actuator, an interference/grinding noise was heard and the actuator stopped functioning properly. The team disassembled the Urs designed actuator and noticed that the planetary gears had sheared, as shown in Figure 6 below.

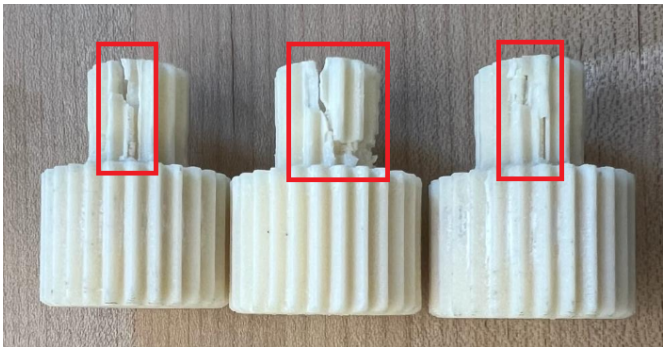


Figure 6: Sheared Planetary Gears

After conducting a root cause analysis, the problem was separated into two factors: 1) insufficient strength in the material choice of standard PLA filament, and 2) Insignificant infill percentage. To combat these problems, we reprinted and replaced the planetary gears after retrieving the Computer Aided Design (CAD) files from the client's given dropbox. We then proceeded to print on the ONYX printer, which uses material made up of micro carbon fiber filled nylon for 71 MPa of flexural strength compared to 50 MPa of standard filament [5]. Additionally, we specified the infill to be increased from the default 10% to 40% when reprinting the gears at the Myhal Fabrication Facility. The resultant gears can be seen in Figure 7 below.

Figure 7: Revised Planetary Gears

Although the gears were manufactured more robustly, this does not eliminate the possibility that they will shear again. In order to decrease the likelihood of recurring shear, future teams working with this actuator should spend more resources in the manufacturing process (i.e. use SLA, better slicing profiles, better print parameters).

2.2.2 Housing Stripping

During reassembly of the actuator, we attempted to screw in the connecting bolts to the housing, but were unable to securely fasten the housing due to screws being loose as a result of screw threads of the housing holes stripping. Unsecure fastening of the actuator housing poses risks including component misalignment, electrical hazards, mechanical failure, and operational inaccuracy. The team realized that the actuator was designed for a one-time assembly with thread forming screws into the plastic housing; it was not designed to be disassembled. We explored the option of utilizing heat set inserts as a removable assembly option to the housing by testing the heat set inserts on another printed gearbox. The headset inserts could not be properly inserted since the smallest available headset inserts were too large in diameter for mating holes. Avenues to explore for future teams would be to enlarge the hole diameters to allow for removability of the component parts. Unfortunately, the team did not have time to make this diametrical adjustment, instead a new housing was printed (shown in blue filament) with thread forming fasteners, recognizing that it is not a permanent solution. The assembled actuator is shown below in Figure 8.

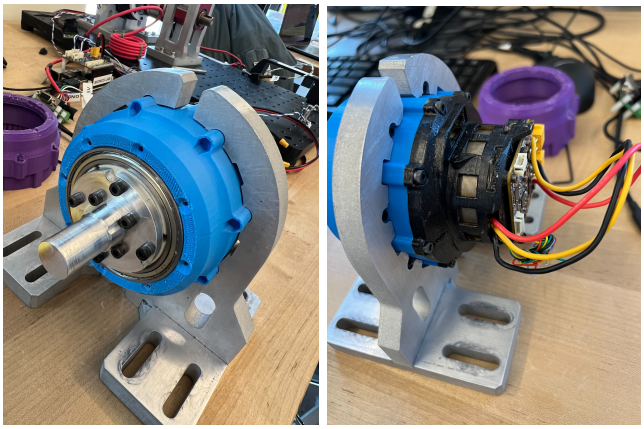


Figure 8: Final Housing Assembly, Left - Front View, Right - Back View

To ensure all of the necessary components are available when reassembling the actuator, the team put together a full Bill Of Materials (BOM) of the actuator to cross reference and place shipment orders of any missing components necessary for assemblage. The BOM shown below in Figure 9 was based on the given T-Blue BOM and actuator CAD given in the client's dropbox.

	A	B	C	D	E	F	G
1	Component Category	Component	CAD Link	QTY	Vendor Links	Price	Comments
2	3D Printed Parts	ASA_P12_LONG_release	ASA_P12_LONG_release.stl	3	MyFab	$9.27	Planetary gears
3		ASA_R1_LONG_release	ASA_R1_LONG_release.stl	1	MyFab	$6.93	Ring gear
4		ASA_R2_LONG_release	ASA_R2_LONG_release.stl	1	MyFab	$14.09	Secondary ring
5		ASA_WOLFROM_OUTPU	ASA_WOLFROM_OUTPU...	1	MyFab	$27.21	Outer housing w
6		ASA_WOLFROM_PC_INP	ASA_WOLFROM_PC_INP...	1	MyFab	$4.63	Looking similar i
7		ASA_WOLFROM_PC_OU	ASA_WOLFROM_PC_OU...	1	MyFab	$4.68	Same as above
8		ULTEM1010_S_ROTOR_L	ULTEM1010_S_ROTOR_L...	1	MyFab	$4.48	Sun gear
9		ULTEM1010-RI50_PLANE	ULTEM1010-RI50_PLANE...	1	MyFab	$29.51	Motor housing w
10	Hardware	M3 Locknut	M3 Locknut: 90576A102_...	1	https://www.m	$ 4.65	packs of 100
11		M3 x 40 Bolt	M3 x 40 Bolt (maybe): 912...	1	https://www.m	$ 9.77	had to cut ther
12		M2 Nut	M2 Nut: 90592A075_Steel ...	1	https://www.m	$ 2.35	packs of 100
13		5x50mm Dowel Pin	5x50mm Dowel Pin: 91595...	1	https://www.m	$ 8.23	packs of 10
14		M3x12 SHCS	M3x12 SHCS: 91290A117...	1	https://www.m	$ 10.85	packs of 100
15		M2.5x10 SHCS	M2.5x10 SHCS: 91290A10...	1	https://www.m	$ 10.43	packs of 50
16		M2x8 SHCS	M2x8 SHCS: 91290A015_...	1	https://www.m	$ 18.04	packs of 100
17		MR30	N/A	1	https://www.a	$ 20.00	10 male/femal
18		30mm x 10mm cooling f	N/A	1	https://www.a	$ 10.00	2 in pack
19	Actuator Electronics	T-Motors R150 KV100	N/A	4	https://store.tr	$ 61.90	
20		moteus r4.8 developer k	N/A	1	https://mjbots.	$ 214.00	The dev kit inc
21		mjbots power dist r4.3b	N/A	1	https://mjbots.	$ 139.00	PDP
22		moteus r4.8	N/A	3	https://mjbots.	$ 104.00	just the contro
23		mjbots pi3hat r4.4b	N/A	1	https://mjbots.	$ 149.00	Pi daughter bo
24		Bearings	N/A				
25		6702zz Bearing	N/A	1	https://www.a	$ 15.00	
26		688zz	N/A	1	https://www.a	$ 20.00	
27		6704zz	N/A	1	https://www.a	$ 18.00	
28		105zz	N/A	1	https://www.a	$ 16.00	
29		6810zz	N/A	1	https://www.a	$ 9.49	
30	**Total**					**$941.51**	

Figure 9: Urs Actuator Bill of Materials

Not all of the components needed to be reordered, but this BOM served as a good checklist to know what was missing and how much it would cost.

2.2.3 Snapped Motor Coils

On March 4th, during the course of the experiments, the driven actuator motor stator experienced damage to the coils, ranging from significant surface wear to complete snaps on two of the coils. The cause of this failure is theorized to be from friction between the stator coil and planet carrier (PC) caused by sliding of the PC relative to the bearing, eventually leading to the wear and rupture of the copper wires.

2.2.3.1 Failure Investigation

While running the testing procedure with the two motors connected via the torque shaft, the driven motor (motor 4) suddenly stopped responding to commands after a few successful velocity control runs, despite the drive motor (motor 1) running in velocity/position/torque control without problems. The lack of acoustic noise, along with the stationary condition of the output shaft, indicated that the actuator was not running at all; if there had been a transmission issue, the team would have observed vibration (noise, mechanical, heat, etc.). The dynamometer was taken apart and the problematic motor (motor 4) was run in velocity/position control independently. Moteus debug UI showed consistent and correct readings from motor 4 encoders when the motor was turned by hand. Because the voltage and temperature values were standard, the electronics were assumed to be functional and the investigation was directed towards the assembly of the actuator. To understand the root cause of failure, the team went back to [2] to understand the intended function.

The actuator was taken apart to identify the source of failure, and it was found that the planet carrier was making contact with the motor stator. In the diagram below, the component responsible is the left half of the planet carrier input (picture E, grayish pink part labeled PC). According to the diagram, the components stopping the left half of the PC from moving translationally towards the motor stator (MS) are the tight fit with the outer ring of the RSG connecting bearing and the screws fastening the two halves of the planet carrier. Section 2.2.3.3 explains the discrepancies between the intended design and assembly process followed by the previous team, which ultimately led to the failure of the actuators.

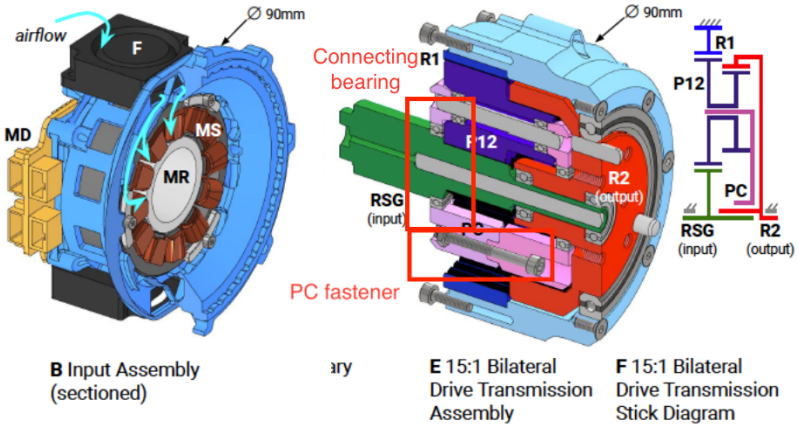


Figure 10: Diagram of Actuator Assembly from “Design and Characterization of 3D Printed, Open-Source Actuators for Legged Locomotion” [2]

2.2.3.2 Evidence for Planet Carrier Displacement

Upon disassembly, the left half of the PC was found to have translated towards the MS coils, as can be seen in Figure 11. The right picture shows a groove around the shaft with residues of copper on the bottom of the PC, indicating contact with the MS coils.

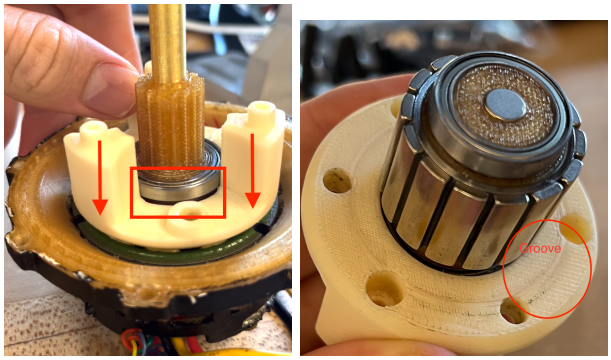


Figure 11: Left - Displacement of the Planet Carrier (PC) Relative to the RSG Connecting Bearing, Right - Signs of Contact between PC and Motor Stator (MS) Coils, 2024-03-04

The damage to the coils can be observed in Figure 12, areas of significant wear are circled and wire ruptures are boxed out. This damage was sufficient to cause complete failure, which was reflected by the non-response to microcontroller inputs.

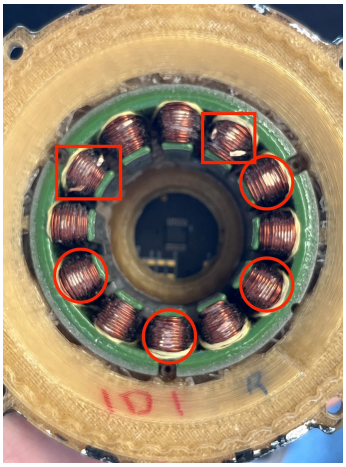


Figure 12: Wear and Failure of MS Coils, 2024-03-04

2.2.3.3 Potential Explanation for Failure

While taking the actuator apart, it was found that the two halves of the planet carrier were not fastened as intended in Figure 10. Figure 13 shows the missing bolt and screw connections that allowed for contact between the planet carrier and motor stator coils. The tight fit with the RSG connecting bearing was also observed to be looser. The reason for their separation is likely motor vibration over time.

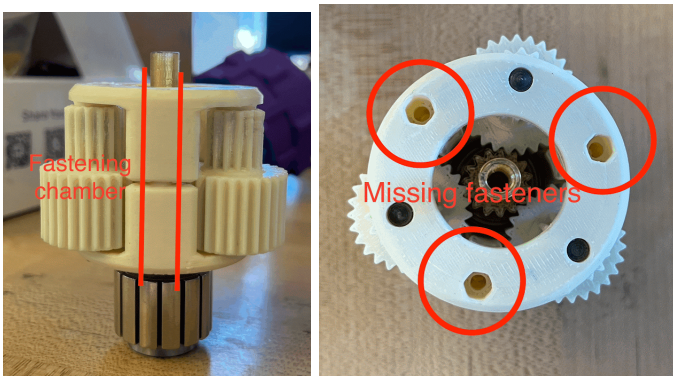


Figure 13: Left - Fastening Chambers that Keep Both Halves of the Planet Carrier Together. Right - Missing Fasteners Upon Disassembly, 2024-03-04

The team tried to fasten the two parts while re-assembling the actuator, but it was found that the holes for the screw and nut were not big enough. Likely, the CAD for the planet carriers was not modified to 3D print with enough clearance for fasteners. It is assumed that, to save time, the previous team neglected these fasteners and relied on the tight fit tolerance to hold the PC halves in place. Figure 14 shows the tolerance mismatch.

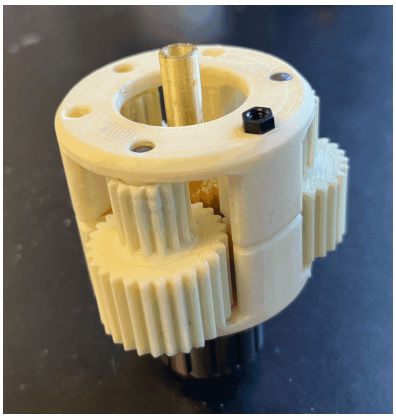


Figure 14: Tolerance Mismatch between Bolt and 3D Printed Planet Carrier, 2024-03-04

To fix this issue, the team chose to use threaded inserts in the fastener holes. With the use of a soldering iron, the inserts were heated to melt the plastic; once in place, they were held while the plastic hardened. This temporary solution allowed for easy assembly and disassembly, while providing protection for the motor stator from the PC. In the future, the actuator either needs to be redesigned or the assembly process needs to be standardized and enforced.

2.2.3.4 Concerning Areas of Failure

Despite finding a temporary fix, the motor stator coils are still at risk of failure with the current design and assembly process. While investigating reserve actuators, it was noticed that many of them had similar coil damage, but not to the point of failure. A few of these actuators were assembled as intended; however, the discrepancies in tight fit tolerances from 3D printing, coupled with the fatigue from usage and vibration, allow the components to move relative to one another. Based on these findings, four modes of motor stator

failure were identified in Figure 15 and ranked by severity: I) the gear (P12 in Figure 15) press-fit rods get loose and come in contact with the motor coils, II) the PC halves separate and the lower half rubs against the motor coils, III) the PC fasteners come loose and hit the motor coils, and IV) the whole PC shifts down and hits the motor coils. If the actuator is assembled properly, the only real concern for failure is II) as the current design does not stop the rods from translating towards the motor stator when vibrating (it only remains in place through a tight fit with the PC). As was seen in this section, mode II) can happen if the PC halves are not fastened properly. Mode III) may happen over an extended period of time, or if the actuator gearbox is subject to enough heat to melt the plastic; washers can be used to keep the fasteners in place. Mode IV) is unlikely to happen: the gears would need to fail completely, and the bearing connecting the PC to the output shaft would need to be extremely loose. To eliminate Mode IV), the entire actuator gearbox would need to be manufactured from stronger materials (steel components), but it would come in at a significant cost.

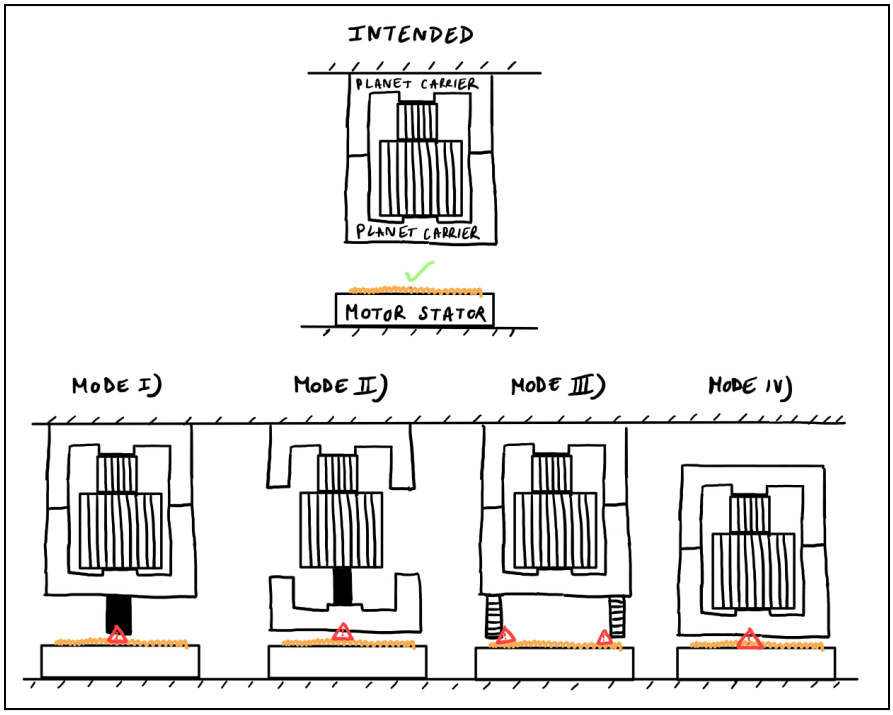


Figure 15: Intended Placement of PC Assembly and Various Modes of Failure

2.3 Electrical System

The electrical diagram of the Benchtop can be seen in Figure 16. The design has two main power sources, which can be somewhat classified into isolated high voltage (HV), and low voltage sources.

The high voltage supply is supplied by two LiPo batteries, each with a nominal voltage of 22.2V (6S cell Clientconfiguration). The discharge rate of each LiPo is 5C [6], giving a max amperage of 165A continuous. In the provided documentation for the benchtop, it is mentioned that the batteries are only charged to 75% capacity to “ensure sufficient headroom for regeneration” [6]. The two batteries are connected in parallel, so the resulting bus voltage to the power distribution boards is equivalent to the

nominal voltage of each LiPo battery, but the current load on each battery is halved compared to using a single battery. No cell balancing is employed during operation of the Benchtop mechanism. A power distribution board (PDB), the mjbots power dist r4.3b, is used to distribute power to both motors in the testing setup. This board provides pre-charging and breakout for power connections (via XT30 connectors) to up to 6 motors. Both motors have identical electrical layouts, with the output power from the PDB being fed through an inline power meter (INA260). The INA260 measures current and voltage drawn by the controller board on each actuator, and transmits this data via I2C to the test computer. Finally, the motors are controlled with mjbots Moteus r4.11 controllers, which is a complete package including drive electronics (enabling both forward and backwards power delivery), magnetic shaft encoder, high performance 32 bit microcontroller, and high speed CAN interface.

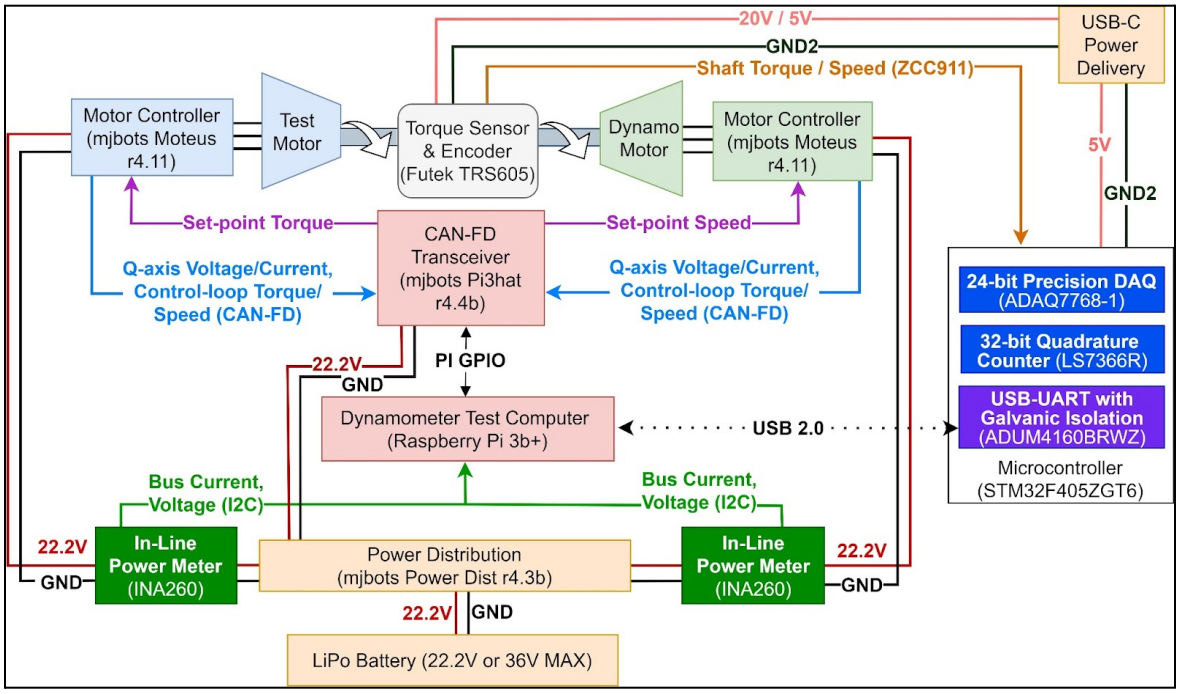


Figure 16: Electrical Diagram of the Benchtop [6]

On the low voltage side, a Raspberry Pi 3b+ (RPi) is used as a primary test computer. Mounted on the RPi is a CAN-FD transceiver hat (mjbots Pi3 Hat r4.5), which enables CAN-FD communication with the motor controllers, as well as regulated power input from the batteries. The power regulation from the pi3hat is important for the Pi 3b+ to receive isolated power from the battery rather than a wall power supply. One caution that the team took was to ensure the Pi only received one source of power (since it is also possible to power the RPi with the 5V micro-usb as well as the Pi3 Hat). Torque and speed values are captured by the torque sensor (TRS605), mounted on the shared motor shaft. Torque values from the TRS605 are processed by a 24-bit DAQ (Digital Acquisition) unit into a digital value. Velocity values from the TRS are fed into a 32-bit Quadrature counter. Both microchips (for torque and velocity) are connected to a STM32 microprocessor via SPI. Ultimately, the torque and velocity data is transmitted to the primary test computer (RPi) via an isolated UART interface.

Similar to the mechanical system, the electrical system was fully procured and assembled at the beginning of this project. However, the system broke down at various components throughout the project, requiring engineering effort to repair and improve. In the following sections, we describe all sources of failure observed during the course of the project.

2.3.1 Torn JST on Power Distribution Board

On March 1st, the JST that connects to the global power switch on the Power Distribution Board (PDB) was torn off halfway and two of the copper pads peeled off with it. None of the high voltage circuit components were getting power because of this. The broken copper pads were kept and an attempt to solder the JST back on was made, but it was difficult as the copper pads had fallen off the PCB trace. The soldering job worked as a temporary solution, and we anticipated further failure.

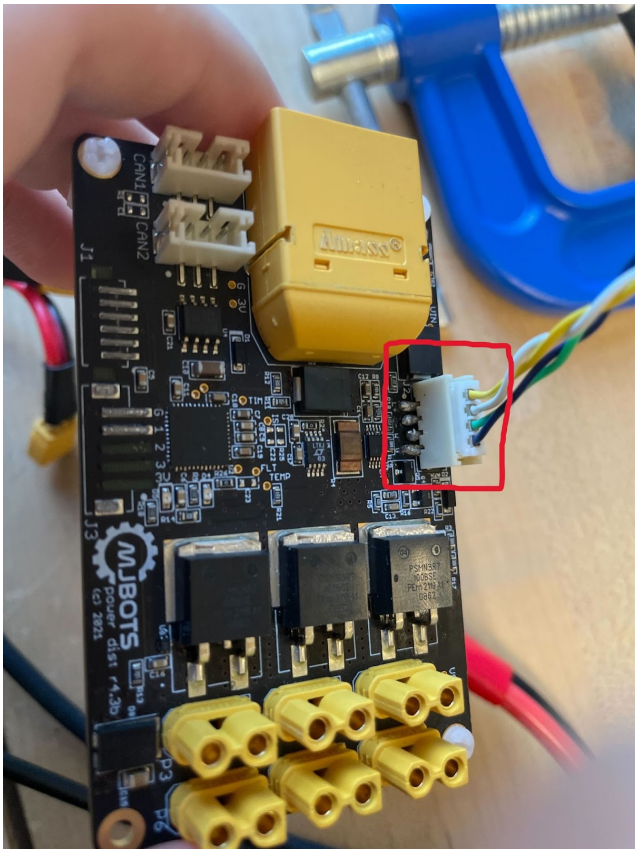


Figure 17: Failed JST Component Highlighted in the Red Box, the Middle Two Pins have Copper Pads Torn Off from the Board with the JST, and are Visibly Bent

On March 23rd, none of the power indicators on the PDB, Pi3Hat, RPi, controller bus or controllers would turn on, indicating an issue with the PDB. Upon investigation, the solder points connecting the switch, highlighted in Figure 17, had failed. Specifically, the third solder point from the left had lost all connection to the PCB board. To fix this, the JST was first desoldered. The area underneath the JST connector, highlighted in Figure 17, had failed entirely as the copper traces completely peeled off from the PCB, preventing further repair. One possible solution was to scratch off the solder mask, exposing the wire below, and soldering directly onto the wire in the PCB. However, this is quite risky and could damage the

PCB itself. Thus, connections were soldered using a new off-board JST connector to equal node connection points on the PCB. Electrical tape was placed over the exposed wires. This is not a permanent solution, and should result in complete replacement of the PDB in the future.

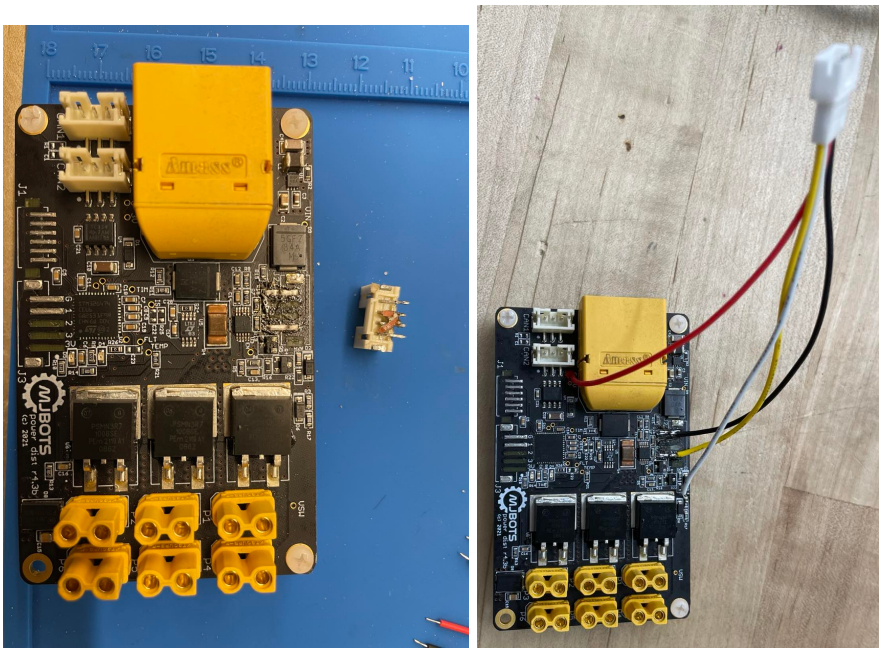
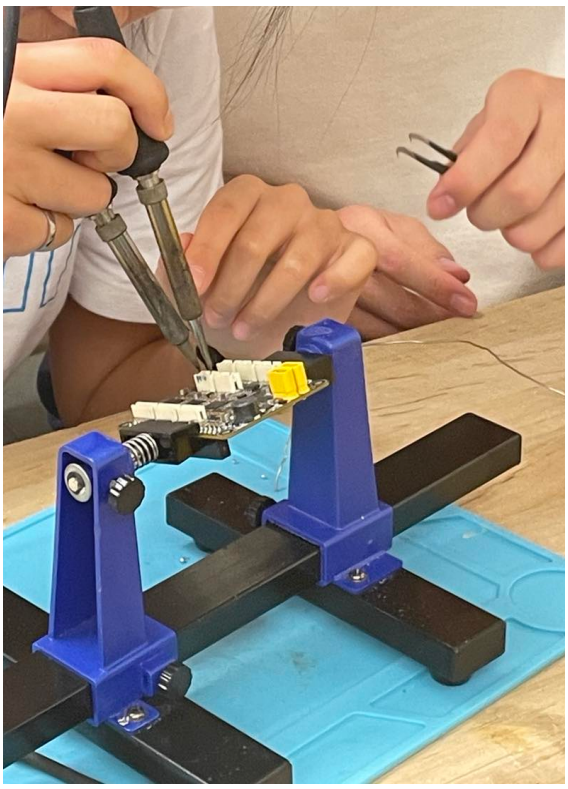
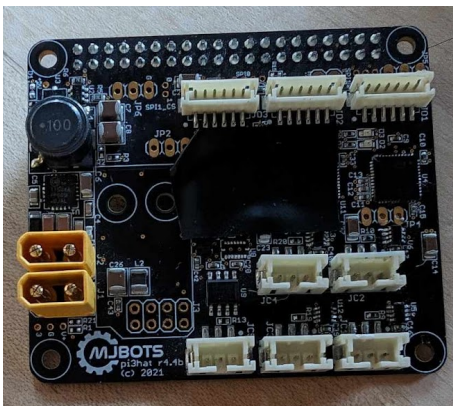


Figure 18: Left - After Desoldering, Right - After the Makeshift Solution

2.3.2 Issues with Pi3Hat

On March 10th, after a few successful runs of the dynamometer setup, the controllers suddenly could not be detected on TView or in the console. The cause of the failure was a faulty CAN transceiver chip, which was diagnosed by checking the termination resistance of the CAN channels on the chip. In three of the five available CAN ports on the Pi3 Hat, the termination resistance for both CAN-H and CAN-L (the two channels of the CAN bus communication protocol) was less than 10KOhms. This behavior is indicative of a chip failure, and results in the inability to send messages via the transceiver. Moteus support suggested resoldering the CAN transceivers, or ordering a new Pi3 Hat board. Both approaches were attempted.





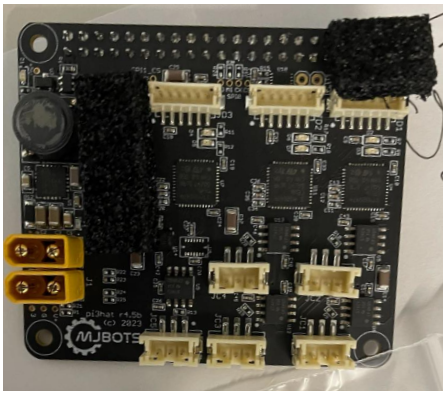


Figure 19 a, b, c: Left - Soldering Transceivers with 2 Soldering Irons.

On March 20th, the Pi3Hat stopped receiving power, as indicated by the power LED on both the RPi and the Pi3 Hat. At this point, it was safe to assume that either the cause of previous failures or prior soldering attempts had damaged a crucial component of the board. A new board, the Pi3Hat r4.5b, was ordered. This board also had the unintended benefit of housing a different CAN transceiver model, ones with 60V fault protection, making them more resistant to the previous failures for these transceivers. The new board worked with the rest of the electrical system with no problems.

2.3.3 Decoupling Motors and Controllers

As a preventative measure to future motor failures, improvements were made to enable motor replacement in the case of failure.

Originally, each motor was soldered to a controller by wire, which made it hard to mix and match controllers and motors. To decouple them, we cut the wires and soldered on HRC Tri-pole connectors for 2 motor-controller pairs (one on March 18th, one on March 20th). This solution was implemented to make future troubleshooting more accessible and efficient.

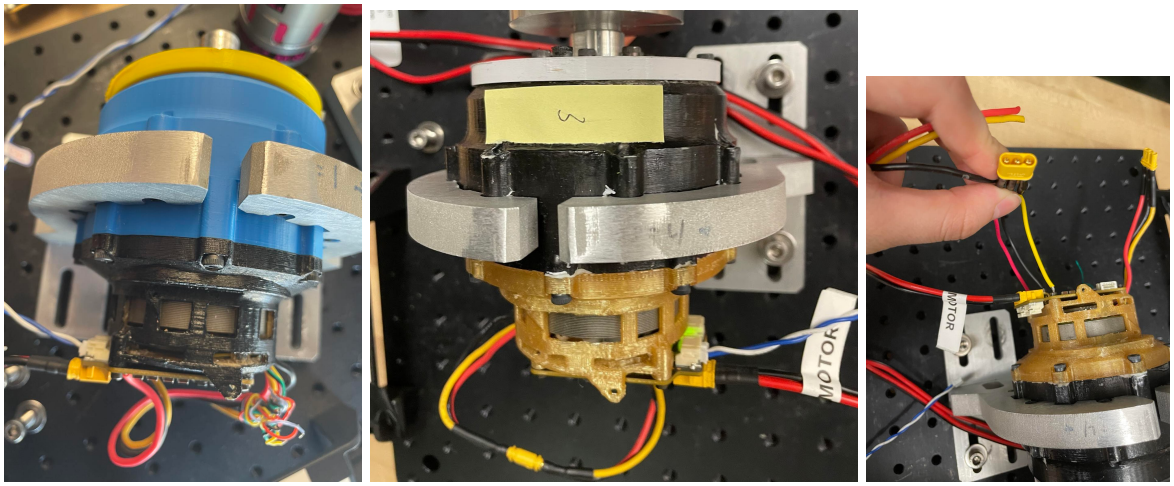


Figure 20: Left - Before Adding Connectors, Center - After Adding Connectors (Bottom), Right - Closer View of Connectors - Yellow Wire Goes to the Single Pin in the HRC Tri-Pole

2.4 Software

2.4.1 Architecture

The source code used in this project consists of a C++ codebase for automating the testing sequence, and some Python scripts for post processing and real-time GUI display. On a high level, the C++ code takes as input a configuration file which specifies motor parameters such as Pi3Hat port and controller mode, as well as a comma-separated file, “test.txt”, where each row specifies one operating point on the efficiency graph. The code then iterates through the rows of “test.txt”, going through the testing sequence for each point – ramp up velocity, ramp up torque, hold, then ramp down. During this process, it generates 3 sets of logs: 1) “logs.txt” - general logs for software states and debugging, 2) “motor_logs.csv” - CAN results from Moteus controllers which yield different measurements of torque/velocity than the torque sensor, 3) “sensor_logs.csv” - measurements from INA260 and torque sensor. The raw data from these logs feed into two Python scripts, one for visualizing the torque and velocity as the tests are running, and one for post-processing the sensor data and calculating its corresponding efficiency point on the efficiency graph.

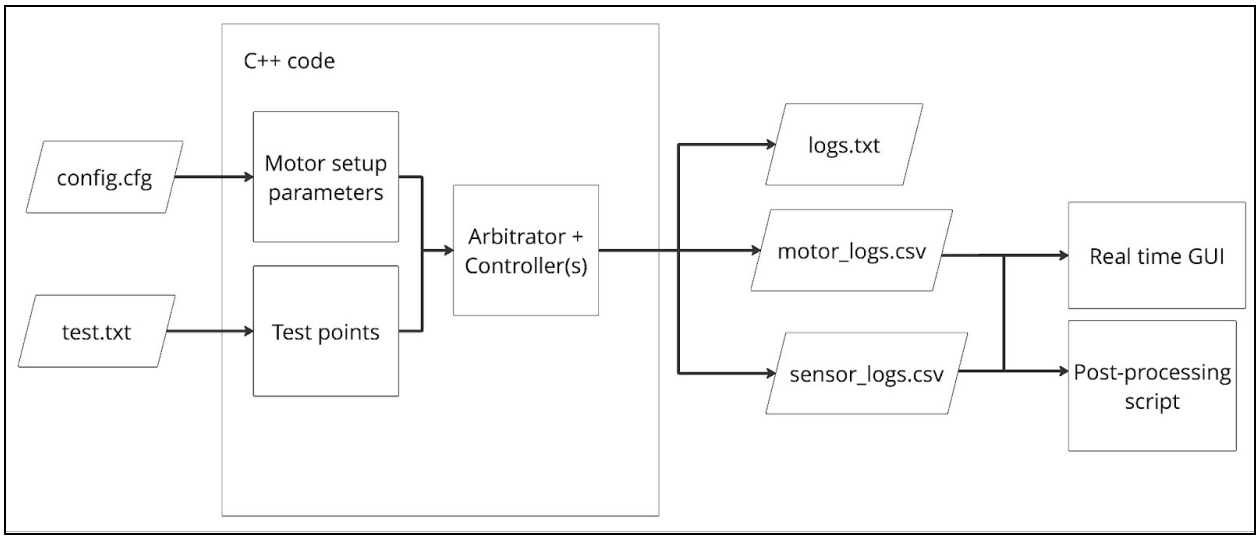


Figure 21: Input/Output Diagram to Software Architecture

2.4.2 Data Collection C++ Code

The original testing code for the benchtop was partially ported from Python to C++ to improve execution performance. In this project, we not only completed the code migration to yield a fully functioning dual-motor testing system in C++, but also improved the performance, readability and robustness of the code.

The C++ code inherited from previous projects works to control a single motor, log its response over the CAN bus, and log the sensor data in a separate file. The Moteus library, which serves as the motor interface, implements a PID position control over an inner FOC (Field-Oriented Control) controller, providing position/torque/velocity control modes by setting k_d, k_i, k_p of the PID controller and parameters of its position control commands. The Moteus library was used in conjunction with the Moteus hardware suite, consisting of a Raspberry Pi, a Pi3Hat, a power distribution board, and motor controller boards. Concurrently, sensor data originating from the Futek torque sensor and power meters connected to each motor were recorded and sent to a custom logger. This logger class was responsible for writing any and all values to recorded log files, stored on the computer hard drive, for further analysis.

On top of fixing various issues in the original code, we reorganized the architecture to be more readable and maintainable, and extended its functionality to control two motors with proper synchronization between threads and correct control commands. We also wrote post-processing scripts in Python to process the raw data collected from motors and sensors, and generate efficiency graphs.

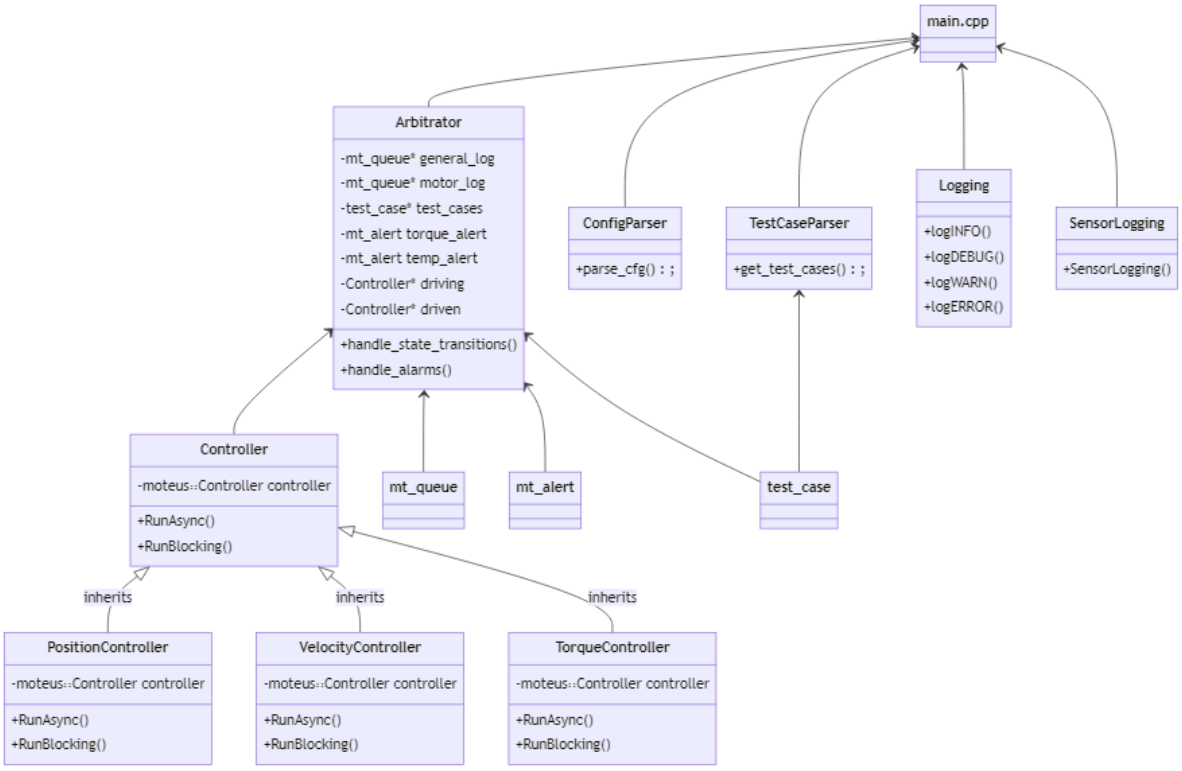


Figure 22: New Architecture Diagram Presented as UML, A->B Indicates A Depends on B

Our current source code is available at https://github.com/iatsl/Benchtop. Please contact the team for access.

2.4.2.1 Existing Codebase and Coding Challenges

While the original codebase laid significant groundwork for motor control and sensor logging, significant additions had to be made to enable energy efficiency testing. First, the controller class had to be updated to support more than just position control. Next, using the updated controllers, a main program had to be written to enable efficiency testing. Previously, the code supported moving the position controller to various test points, but they could not be reached correctly nor held for a sufficient amount of time. Lastly, the test command parser had to be rewritten to support velocity and torque commands as well as refactored to use C++ constructs. We also noticed that the original program caused a jittering effect in the motor because of a poorly written control position loop. Lastly, it had synchronization issues which resulted in a zombie thread heating up the motor even after the program was killed.

2.4.2.2 Code Reorganization and Refactor

The overall architecture of the code was overhauled and reorganized for clarity. The original organization was non-standard (eg. header files and implementation files in one folder) and inconsistent, making it difficult to troubleshoot and test. The new organization is as shown in Figure 22. This architecture focuses

on an OOP (Object Oriented Programming) design with better separation of duties along with an introduction of a new single point of orchestration. Not only does this make development easier, since specific controllers (such as velocity or torque controller) can inherit from a base controller class, but it also makes the code significantly easier to read, since the arbitrator and controller classes abstract away the underlying library features.

Various refactors to the source code itself were also made:

- Enums and structs were introduced for better organization.
- The Moteus and Pi3Hat libraries were updated to take advantage of a simpler controller interface.
- Functions were rewritten to be more idiomatic in C++ instead of pythonic. Examples include removing unnecessary constructions and moves of std::map (likely translated verbatim from dicts in Python code), and rewriting functions named “has_next” and “get_next”.
- Utilization of smart memory features like shared_ptrs and unique_ptrs to ensure code safety and avoid segfaults

We also introduced googletest as a unit testing framework to validate individual components and build robustness. As of now, four test files with ~22 test cases have been added. Functionality for generating randomized test data was also added. This gives us confidence to collaborate as a team on a single codebase while preventing accidental changes to working functionality. Documentation was improved all around and detailed development notes were kept to record solutions to common issues.

2.4.2.3 New Feature/Innovations

The main new features and innovations concern the following components:

1. Added torque/velocity controller
2. New Arbitrator enabling control of 2 motors
3. Improved multithreaded logger

New Feature: Torque/Velocity Controller

Previously, the controller implemented in the codebase uses position control, and this resulted in motor jitter as it was stepping rapidly through a sequence of positions, causing a jerky trajectory profile with sudden acceleration and deceleration required at each test point. To create a smoother trajectory profile, we looked into the Moteus library which provides implicit control modes through setting parameters to special values such as 0 or NaN. Moteus implements a cascaded control with a position control PID wrapped on top of an FOC controller. Since torque is directly proportional to current, implementing torque control is equivalent to skipping the outer PID loop and directly adjusting current. For velocity control for the driven motor, Moteus designates the special value NaN for a position command in position control mode to implicitly specify velocity control.

Some problems were encountered in this process, such as initialization of servo parameter max_position_slip required for velocity control. We read the documentation extensively and consulted any example we could find online -- but similar velocity control use cases are scarce and hard to find on the internet. In the end, we reached out to the original developers on the Moteus team directly, and the issue was resolved through a method not mentioned in the documentation.

Many steps were taken to make our code more maintainable and easier to read for future developers. All controllers are now derived from a base controller class using inheritance. In addition, the previously implemented controller has been updated and kept as the default position controller. The program has been refactored to control two motors instead of just one, and allows the control mode to be separately specified for each motor in the config file.

Compared to previous implementations, this results in easier development when creating a new controller, since certain underlying functions are already exposed by the base controller. Moreover, the base controller specifies virtual functions for running and stopping which must be defined in the derived controllers -- in simple terms, we can use the same functions for all controllers, with the only difference inherent to the specific controller the function is called on. This allows for changing the controller easily depending on the specific use case, as well as making the code much easier to read.

New Feature: Arbitrator

In order to properly conduct the experiment as described in **Section 2.1**, some coordination must exist between the two motor controllers (driving and driven) to enable reaching each torque/velocity test point. An arbitrator, which coordinates multiple components of a system, has been added for this purpose. The arbitrator takes a list of test_cases, provided by the TestCaseParser class, and provides commands to both controllers. Additionally, a finite state machine (FSM) is implemented to programmatically implement the testing procedure states. The FSM is a simple method to encode states and state transitions, which is needed to program the testing process as shown in Figure 4. The states of this FSM are shown in **Table 1**. Additionally, transition conditions are defined as transitions between FSM states. In general, this FSM is fairly simple, with all states proceeding to the next state upon transition conditions being met. However, the STOPPED state is reachable from all states, and occurs when the TestCaseParser cannot provide any more test points to Arbitrator, or an anomalous activity occurs (such as exceeding temperature or torque limits).

Table 1: Arbitrator Finite State Machine states

State	Description	Transition Condition
STOPPED	Motors are stopped. Also functions as an e-stop state.	
TEST_START	Beginning test	Controllers ready New test_case exists
DRIVEN_RAMP_UP	Ramp driven motor to target velocity	Driven motor velocity reaches target velocity
DRIVEN_HOLD_ASCEND	Hold driven motor velocity for set time	Wait X* seconds
DRIVING_RAMP_UP	Ramp driving motor to target torque	Driving motor torque reaches target torque

TEST_POINT_HOLD	Hold driving, driven motor at targets	Wait Y** seconds
DRIVING_RAMP_DOWN	Ramp driving motor to 0 torque	Driving motor reaches 0 torque
DRIVEN_RAMP_DOWN	Ramp driven motor to 0 velocity	Driven motor reaches 0 velocity
TEST_POINT_DONE	Test point complete, record completion and perform calculations	New test_case exists -> TEST_START

*Wait times are set to X=2s by default, but can be reduced in settings
**The duration of TEST_POINT_HOLD state is expected to vary based on test point values for torque and velocity. As such, an exact function for Y (wait duration) is an open task which the team is working on.

The arbitrator was initially designed to operate both controllers (and thus both motors) in parallel, by multiplexing the command instructions such that the underlying motor controllers could operate asynchronously. However, when testing, we realized that this approach did not work. To troubleshoot the complex multithreading issues, we created a simple POC (proof-of-concept) program that focuses on interfacing with Moteus directly, without the OOP architecture boilerplates, to better isolate the point of failure. Through this investigation, we found that the root cause was that the underlying transport mechanism, which relayed instructions from the Raspberry Pi3 to the Moteus controllers, was shared between the two motor controllers. Thus, we decided on a sequential implementation instead, where the two controllers are operated one after another, which is still sufficiently performant.

Innovation: Improved Logger

Previously, the logger instance utilized a polling scheme where the thread running the logger continuously checked the message queues (mt_queue) for available messages. This has significant inefficiencies when no messages are being produced, since the logger is still active and checking for messages. For example, between test points, the logging frequency may decrease to avoid logging points that are not related to the test. As such, the logger should yield computational resources to the other threads during this time.

Additionally, the previous logger only supported two very specific types of log messages, one message for motor data and another for sensor data. Increasing the number of log files, or changing the number of log streams, was impossible as these parameters were hard-coded. The improved logger supports a variable number of logging channels, where each channel can be piped into a new file stored on disk. Each logging channel is run in a thread, which sleeps when the message queue for that channel is empty. A pseudocode implementation is shown in Table 2, which represents the logger improvement in concept.

Table 2: Logger Pseudocode

Old Logger	**New Multithreaded Logger**

```
message_queues = {motor_queue, sensor_queue}
log_files = {motor_log_file, sensor_log_file}

while(true){
        for(queue in message_queues){
                if(queue has message and !closed){
                        logToFile();
                }
        }
}
```

```
message_queues = {queue_1, …, queue_N}
log_files = {log_file_1, …, log_file_N}

for(queue in message_queues){
        run in thread(log_thread, log_file, queue);
}

def log_thread(log_file, queue){
        while(queue has message and !closed){
                logToFile();
        }
        wait until (queue has message);
}
```

The improved logger was able to achieve logging rates of over 8000 logs/sec when 1 logging channel was tested, and over 7000 logs/sec when 2 logging channels were tested. A sample of these results is shown in Figure 23.

```
[ RUN      ] LoggingTest.RateTest
attempting to register log mt_queue to log file ../test/test_data/rate_test.log
registration complete
starting logger on 1 threads
closing logger threads
all logger threads closed
logging 25000 samples took 2.93183 seconds
[       OK ] LoggingTest.RateTest (2941 ms)
[ RUN      ] LoggingTest.RateTestDouble
attempting to register log mt_queue to log file ../test/test_data/rate_test_d1.log
registration complete
attempting to register log mt_queue to log file ../test/test_data/rate_test_d2.log
registration complete
starting logger on 2 threads
closing logger threads
all logger threads closed
logging 25000 samples took 3.1797 seconds
[       OK ] LoggingTest.RateTestDouble (3199 ms)
```

Figure 23: Improved Logger Rate Testing Results

2.4.3 Post-processing scripts

There are two post processing scripts which process the data obtained during the testing procedure, specifically during the “hold” stages of each test case. For the Graphical User Interface (GUI) both the motor log and sensor log are used, while for calculating the efficiency map, only the sensor log is used. Table 3 and 4 shows the columns of interest in the motor log and sensor log respectively and what they represent. The motor log gets its information from the Moteus controller while the sensor log gets its information from the torque sensor and power meter.

Table 3: Description of the motor log columns

Column Name	Description	Units
time	Time elapsed	seconds
drive_velocity	Moteus controller velocity for driving actuator. The velocity reported is that after the gearing system	Hz
drive_torque	Moteus controller torque for driving actuator. The torque reported is that after the gearing system	Nm
driven_velocity	Moteus controller velocity for driven actuator. The velocity reported is that after the gearing system	Hz
driven_torque	Moteus controller torque for driven actuator. The torque reported is that after the gearing system	Nm

Table 4: Description of the sensor log columns

Column Name	Description	Units
time	Time elapsed	seconds
pwrmtr_drive_voltage	The voltage of the drive motor from the power meter sensor (INA260)	Volts
pwrmtr_drive_current	The current of the drive motor from the power meter sensor (INA260)	Amps
pwrmtr_driven_voltage	The voltage of the driven motor from the power meter sensor (INA260)	Volts
pwrmtr_driven_current	The current of the driven motor from the power meter sensor (INA260)	Amps
sensor_torque	The torque registered by the torque sensor at the output shaft	Nm
sensor_counts	Number of Rotations experienced by the shaft. This can be converted to velocity by multiple by $2\pi/time_{elapsed}$	revolutions
arbitrator	Identifies which section of the test procedure the data point was on	/

Below in Figure 24 is a GUI showing how the driving velocity and the driven torque behave throughout the testing stages of velocity ramp up, torque ramp up, test point hold, torque ramp down, and velocity ramp down for the duration of one test run.

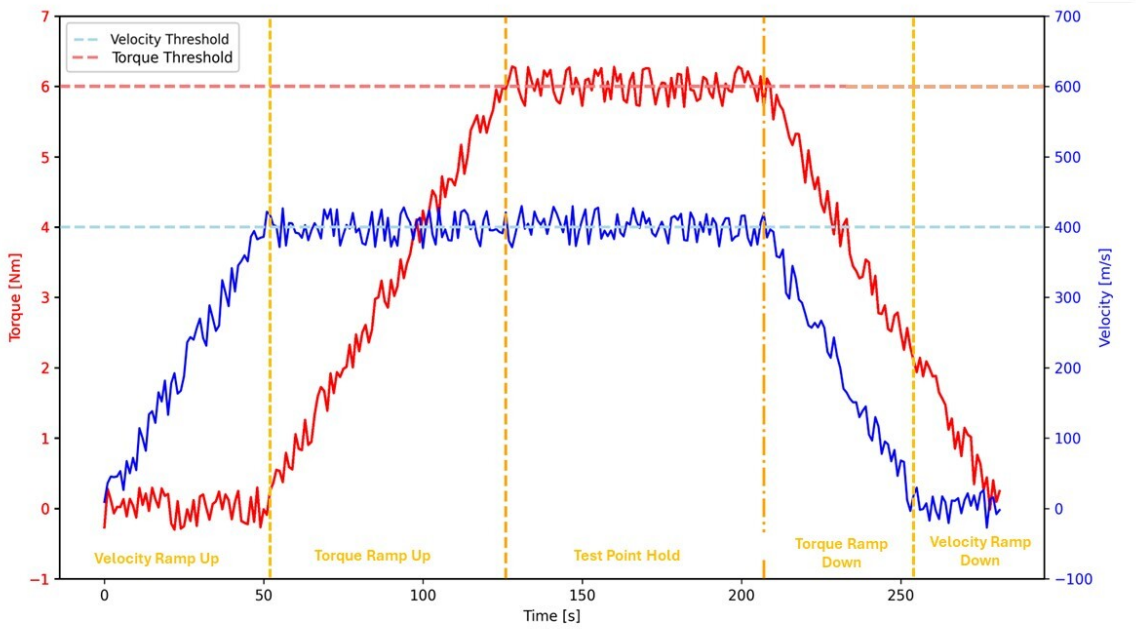


Figure 24: Actuator Torque and Velocity Behavior with Respect to Time

This graph is based upon randomly generated data that are within set torque and velocity limits based on the motor and sensor logs. On the day of the presentation, this GUI will take inputs from real-time data being outputted by the dynamometer for a more realistic dynamic graphical output.

To calculate the efficiency for each test, a specific process must be followed. We first need to obtain all data points where the arbitrator is at the "hold" state, and take the mean over that period. The sensor count data is converted to rad/s by multiplying by $2\pi/time_{hold}$. Once these are done, the efficiency can be calculated. To calculate efficiency we need to determine which quadrant we are in. In the two quadrant case (Quadrant I and IV), velocity is held positive while torque can be either positive or negative. In this case, regeneration only occurs when the torque value registered is negative. Once the quadrant determined, the efficiency can be calculated with the following equations:

$$efficiency = \frac{T\dot{\theta}}{IV} \quad \text{load driven by motor, quadrant I (3)}$$

$$efficiency = \frac{IV}{T\dot{\theta}} \quad \text{motor driven by load, regen, quadrant IV (4)}$$

For calculating efficiencies in Quadrant II and III, velocity is held negative, and the efficiency equations are swapped relative to Quadrants I and IV.

2.4.5 Troubleshooting guideline and common issues

Troubleshooting generally means making a list of potential sources of failure and isolating parts of the system to narrow down the root cause. Since the software system is built on top of electrical and mechanical systems, all else is premised on the electrical and mechanical system working, which means

checking power indicator lights and motor functionality. The Moteus GUI Tview allows for visualization of controller statistics and configurations and provides a console for directly sending controller commands to the Pi3Hat. This makes it ideal for checking functionality of motors upon startup.

In general, the best way to resolve any Moteus related issues is with the Moteus discord. We first searched the discord for keywords to see if it has been addressed before, then started a thread if we had more specific questions.

List of common issues and their solutions:

1. if tview on startup does not dump configuration of motors, either the command is not run with the proper parameters for --pi3hat-config (check which JC port is connected to which motor id), or the motor id is incorrect; when tview starts up correctly, you should see a long list of configs being dumped in the console, and in the top left panel you should be able to see a dropdown from the motor id with various motor/servo statistics
2. if tview starts up correctly and motor does not spin on position control command (see below for proper position control command), try moteus_tool calibration; also try turning the motor by hand and observe in the left panel of dropdowns, *your motor id* > servo_stats > position changes correctly (unit is in revolutions)
3. if tview starts up correctly and servo_stats > position does not change correctly when hand turning the motor, it is an encoder issue
4. if tview starts up correctly and motor spins in velocity/torque control, but does not reach the reference velocity/torque, the controller gains are not configured properly
5. if tview starts up correctly and motor spins slowly with low and loud droning sound
    a. Check the gearbox ratio parameter `conf get motor_position.rotor_to_output_ratio` in tview console
    b. If it's not 0.066849, update it with `conf set motor_position.rotor_to_output_ratio 0.066849`
    c. Do `conf get motor_position.rotor_to_output_ratio` again to check that it is updated
    d. *This is a known issue for motor 1 especially.
6. if using a new motor, need to first find or set the motor id with moteus_tool
    a. Ensure that the motor is connected to JC1 on the pi3hat board, and no other motor is connected to the board
    b. Run 'sudo python3 moteus.moteus_tool --info'. You should see a JSON dump with an id at the start.
    c. If you wish to change the id, run 'sudo python -m moteus.moteus_tool -t 1 --console'
    d. While the process is running, type 'conf set id.id 2' for an id of 2 for example. Ctrl-D to exit,.
    e. Run 'sudo python -m moteus.moteus_tool -t 2'. You should see an 'OK' response from the previous command.
    f. While the process is running, type 'conf write'. You should see an 'OK' response. Ctrl-D to exit.
7. if 'sudo python3 moteus.moteus_tool --info' fails to dump any information or ends with error, check that only one controller is connected to the Pi3Hat and on port JC1; if the connection is correct,

8. if calibration with moteus_tool ends with error “RuntimeError(f”Controller reported fault: (int(servo_stats.fault))”)”, check the battery voltage, it may be too high; you may set servo.max_voltage in moteus_tool --console or tview
9. If calibration with moteus_tool ends with error “encoder not an integral multiple of phase, 0.4761716>0.1”, it is saying that the small encoder at the back of the moteus controller board which interfaces with our motors from a hole on the casing is not placed appropriately – this is most likely a mechanical assembly issue.

3.0 Simulation

The goal of the simulation is two fold. The first one is to obtain similar values as the benchtop to validate the model. The second is to obtain actuator efficiency values at different movement points in the human gait cycle, allowing us to evaluate energy regeneration with a quasi-direct drive motor.

The simulation code can be accessed at: https://github.com/AAImrit/verstraten_model.git

3.1 Simulation Equations

An actuator for a single joint can be represented as a motor connected to a gearbox, which is then connected to a link with a mass (Figure 25). The simulation works by assuming that most of the output variables are known, while the motor values are unknown and need to be calculated.

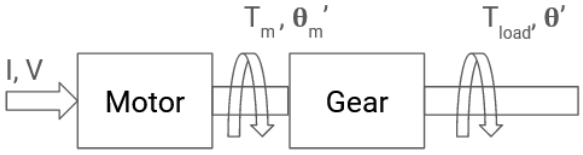


Figure 25: Actuator Broken down into gear and motor component

In [4], it was shown that the energetics of an actuator can be represented through a loaded pendulum with five underlying equations (equation 5 to 9). This pendulum setup applies for the case of a bionic leg: the knee is the joint of focus and the foot experiences the load. The equations are shown below:

$$T_{load} = J_g\ddot{\theta} + v\dot{\theta} + mglsin(\theta) \quad (5)$$

Where, T_{load} is the torque at the output shaft, J_g is the gearbox inertia, v is gearbox damping constant, m is the mass of the joint and link, g is the gravitational constant, l is the link's length, θ is the joint position, $\dot{\theta}$ is the joint velocity, $\ddot{\theta}$ is the joint acceleration. Equation 5 allows the mapping of joint position/velocity or torque to a joint force. The remaining four equations can be used to solve for variables pertaining to the motor.

$$T_m = (J_m + J_g)n\ddot{\theta} + \frac{C}{n} * T_{load} \quad (6)$$

$$\dot{\theta}_m = n\dot{\theta} \quad (7)$$

Where, T_m is the motor torque, J_m is the motor inertia, n is the gearbox ratio, and C is a piecewise function which is $1/\eta_g$ when load driven by motor, and η_g when motor is driven by load (regeneration); η_g is the gearbox efficiency. Equation 6 allows us to calculate motor torque, while equation 7 allows us to map the output shaft velocity (joint velocity) to the motor velocity, which is the velocity before the gearbox stage. The remaining two equations are used to calculate current and voltage, and thus the electrical power of the actuator.

$$I = \frac{T_m + v_m\dot{\theta}_m}{k_t} \quad (8)$$

$$V = L\frac{dI}{dt} + RI + k_b\dot{\theta}_m \quad (9)$$

Where, I is the current input of the motor, v_m is the motor damping constant, k_t is the motor torque constant. For equation 9, V is the motor voltage, L is the motor inductance, R is the motor resistance, k_b is the motor speed constant. Since we are using the same setup as [2], constants were taken from [2] (Appendix B).

Using equations 6 to 9, actuator efficiency can be calculated. There are multiple cases to be aware based on the motor quadrant of operation

$$efficiency = \frac{T_{load}\dot{\theta}}{IV} \quad \text{load driven by motor (10)}$$

$$efficiency = \frac{IV}{T_{load}\dot{\theta}} \quad \text{motor driven by load, regen (11)}$$

There is a third case which occurs at low torque or low speed and high torque when the motor is driven by the load. In this case, the energy from the load is completely dissipated by the motor resistance, resulting in zero efficiency [4, 2].

3.2 Simulation Layout

The simulation is set up to be able to calculate efficiency for both discrete or continuous data and time dependent or time independent data. There are 3 different types of inputs we accounted for: functions of θ or $\dot{\theta}$, discrete time independent values of T_{load} and $\dot{\theta}$ to emulate the benchtop and lastly, biomechanics data which takes in T_{load} and mechanical power. Figure 26 shows the data inputs, parameters for each of the inputs, and the general process of calculating actuator efficiency for each input type.

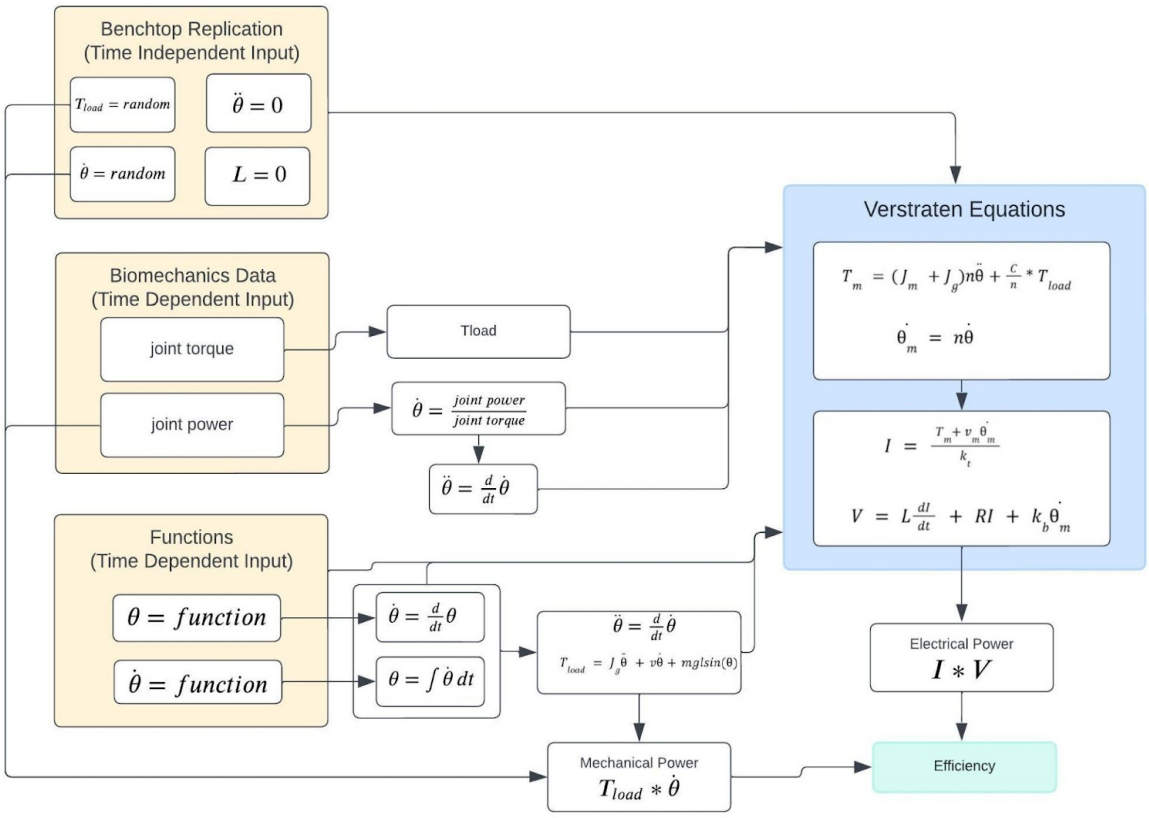


Figure 26: Simulation Layout and Process

3.2.1 Function - First Iteration

The team started by setting up the simulation for a symbolic/numerical function input to replicate and compare our result with [4], ensuring our code is working as expected. The input can either be a function of joint position (θ) or joint speed ($\dot{\theta}$). If the input is θ, the derivative and second derivative is performed to obtain $\dot{\theta}$ and $\ddot{\theta}$. If the input is $\dot{\theta}$, the integral is calculated to obtain θ and the derivative is performed $\ddot{\theta}$. These are then plugged into equation 5 to 10 to obtain the electrical and mechanical power. It should be noted that in cases where numerical differentiation was required, the central difference was used, with forward difference used for the first point, and backwards difference for the last point. The method for calculating efficiency is further discussed in section 3.2.4

3.2.2 Benchtop Replication - 2nd Iteration

The benchtop replication mode aims to obtain similar values as the benchtop test procedure in order to compare the two, validating the model with the benchtop data (ground truth). Similar to the benchtop testing procedure, the input to the simulation is a velocity ($\dot{\theta}$) and a torque test point (T_{load}). Equation 6 to 9 is then used to calculate electrical and mechanical power. The first interaction (section 3.2.1) assumes we have a time dependent input, in other words each data point is consecutive and depends on time. However, when the benchtop is being tested, the test points and values obtained are time independent.

Thus, derivatives and integrals cannot be calculated. To account for this, two assumptions are made. First, since in the benchtop tests the velocity is held constant, the first assumption is that $\ddot{\theta}$ equals 0. We also assumed motor inductance (L) = 0. Typically, L is very small and when multiplied by the derivative of current, the terms $L\frac{dI}{dt}$ in equation 9, it becomes much smaller than the other terms [4]; thus the $L\frac{dI}{dt}$ is often ignored. It should be noted that the saturation limit (the point where the motor stall) is not accounted for in our code.

3.2.3 Biomechanics Data - Final Iteration

The goal of using biomechanics data as an input is to obtain actuator efficiency values throughout the human gait cycle, evaluating the extent of energy regeneration with a quasi-direct drive motor. The biomechanics data was obtained from [7], who collected and processed human locomotion data of 10 different subjects at different walking and running speeds at different inclines. The data provided us with joint power, joint torque and joint position of each subject during different locomotion states/trials. To obtain efficiency for the biomechanics data we used the knee joint torque (T_{load}) and knee joint power ($T_{load}\dot{\theta}$) from [7]. Since the knee joint power equals the joint torque multiplied by the joint velocity, to obtain the joint velocity, we divided the joint power by the joint torque. Numerical differentiation was performed to obtain $\ddot{\theta}$. Equation 6 - 9 is then used to calculate electrical power.

It should be noted that the data provides about 60 human gait cycles for each trial of each subject. In order to calculate the efficiency for one human gait cycle of a specific trial, we first obtained the average joint torque and knee joint power of the 60 cycles available of a single subject. The calculations are then performed with these averages. Moreover, the joint power is given as W/kg and the joint torque is given as Nm/kg. We assumed the kg referred to the weight of the knee joint and link, and thus multiplied them by the "m" constant. Though this is an assumption, through later observation we determine that keeping the per kg would not affect overall efficiency values since this would result in all variables being per kg, including electrical power. When taking the efficiency, the per kg would cancel out.

Originally, we used the the joint position and calculated the $\dot{\theta}$, $\ddot{\theta}$ using numerical differentiation and T_{load} using 5. However, Dr. Laschowki pointed out to us that this may not be accurate since this method does not account for internal moments that are occurring in the human knee system when we are in movement. When we tried validating the T_{load} with the biomechanics data, this was indeed true and our T_{load} was much smaller than that from [7]. Thus, we changed our method to that mentioned above, where we use the joint power and joint torque from the biomechanics data.

3.2.4 Calculating Efficiency

For all of the three input types, the efficiency is calculated with the same method. As shown in equation 10 and 11, during regeneration (quadrant II and IV of motor operation) the numerator is electrical power while the denominator is mechanical power, and the opposite is true for forward motor operation (quadrant I and III). The code first identifies areas of regeneration, using T_{load} and $\dot{\theta}$, in the step where using

equation 6 to calculate T_m, changing the value of C to match the motor quadrant of operation. These same areas are then used when calculating efficiency to decide whether mechanical power or electrical power should be the numerator.

Since we are calculating instantaneous efficiency, areas where there is a switch between the regeneration and forward operation, a discontinuity appears which results in the efficiency bigger than 1 seemingly shooting negative infinity and sometimes infinity. To deal with this, we applied two things. Firstly we capped the efficiency +/- 2, thus for any instance where efficacy is above the threshold, the efficiency is set to +/- 2. We then applied a moving average to smooth out the curves, resulting in efficiency values less than 1, which is the expected values. Since this occurs mainly for negative efficiency values, we expect this to not cause much of an issue since negative efficiency already represents areas where the energy from the load is completely dissipated by the motor resistance, representing zero efficiency [4, 2].

3.3 Simulation Results

Results obtained from the simulation are divided by the input type entered. The section below examines some of the results and insights obtained from each input type.

3.3.1 Function

To ensure the code is working as expected and producing the correct results, the first input and actuator constants were set to be the same as [4]. For these, we set $\theta = 2arcsin(sin(\frac{\theta}{2}) * ellipj(w_p t, sin(\frac{\theta}{2})))$, which is similar to $\theta = sin(t)$. Using this function as the input we obtained Figure 27 a) to c). These results are similar to the [4] which indicates the code works as expected and has been implemented correctly.

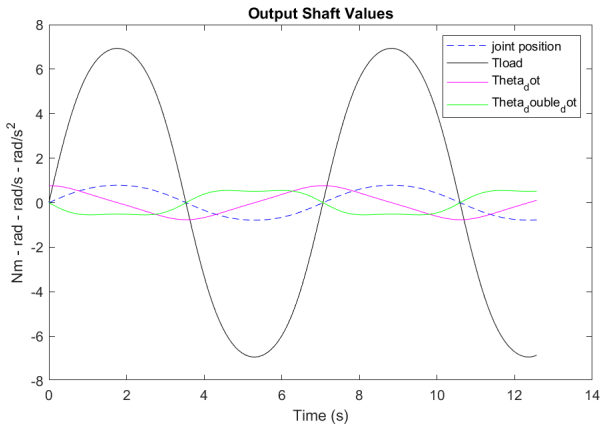
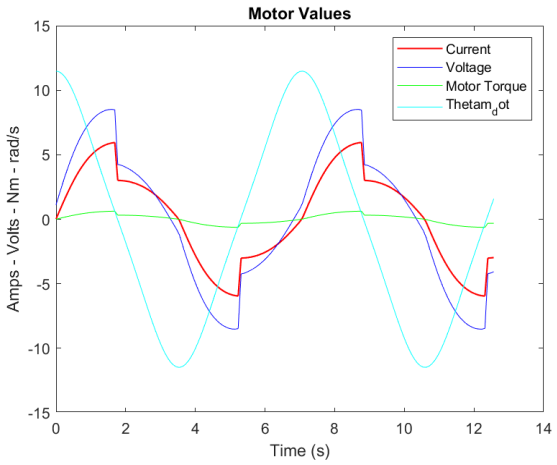
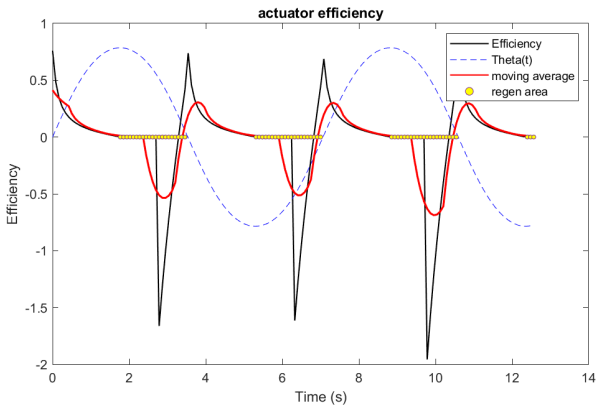


Figure 27 a) Output Shaft Values



b) Motor Values



c) Actuator Efficiency

Notice the jumps in motor torque, current and voltage values. These are discontinuity points which occur when the motor quadrant of operation switches from forward operation to regeneration. These discontinuity points are also seen in [4] and are thus not a cause of concern.

Additionally a step function input for the velocity was tried to see if we can reach a value similar for a velocity-torque point in the [2] graph. This served as proof of concept that the code could potentially replicate the benchtop efficiency map, which would be integral to validating the simulation results. As seen in Figure 28, for a constant velocity input of 1 rad/s, an efficiency of about 0.55 can be reached, which is similar to [2].

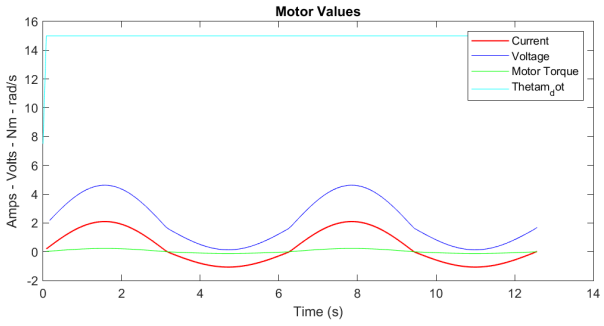
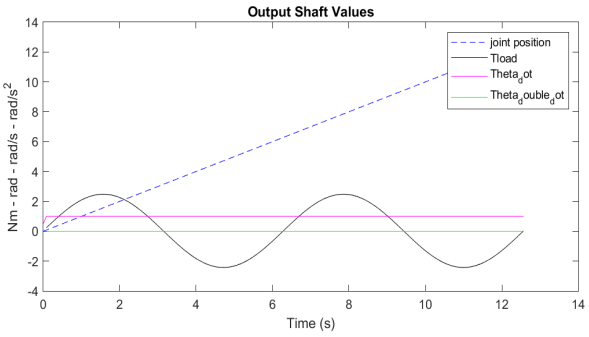
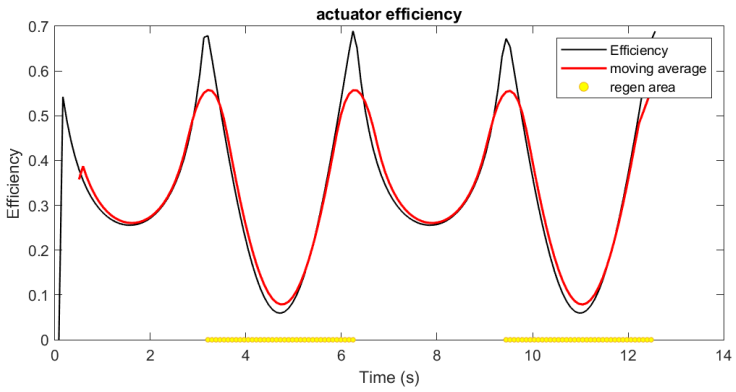


Figure 28 a) Motor Values for $\dot{\theta}$ = Step Input



b) Output Shaft Values



c) Actuator Efficiency

Notice that the efficiency in Figure 28 c) fluctuates in a sinusoidal manner. This is due to the $mlgsin(\theta)$ term in 5 which causes the T_{load}, and thus all other variables, to follow the sinusoidal pattern. Considering this result, the team decided that for the benchtop replication portion (section 3.3.2) the inputs would need to be a constant velocity ($\dot{\theta}$) and a constant torque (T_{load}) to properly replicate benchtop results.

3.3.2 Benchtop Replication

As previously mentioned, from [2], we know the actuator's maximum speed ($\dot{\theta}$) is ~10 rad/s and the actuator's maximum torque (T_{load}) is ~10 Nm. To calculate efficiency for every velocity-torque point, the torque and velocity input are first uniformly distributed, creating a velocity-torque grid. These $\dot{\theta}$ and T_{load} serves as input to the simulation, with current and voltage being the output, which are used to calculate efficiency and create the torque-velocity efficiency map (Figure 29 a)).

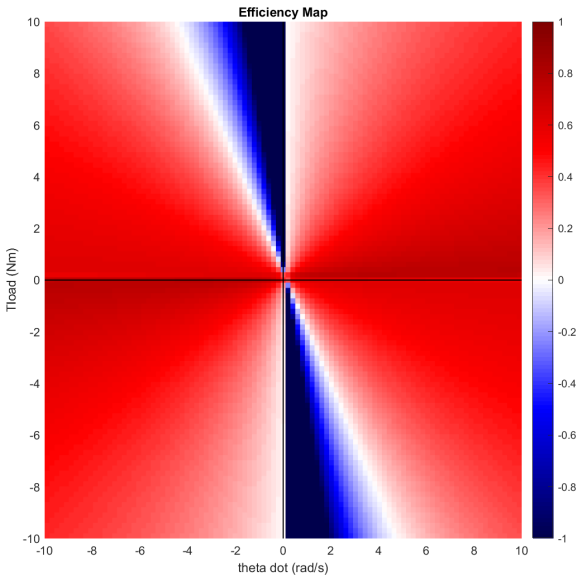
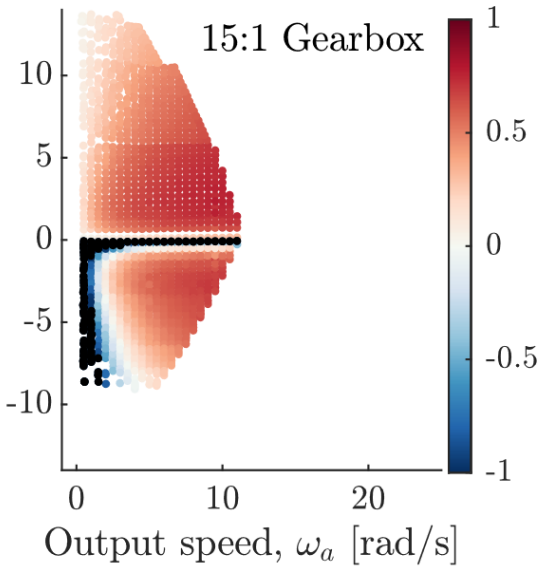


Figure 29 a) Simulation Efficiency Map



b) Efficiency Map [2]

Comparing the simulation efficiency map to Urs' [2] from his physical testing of the actuator, the results look similar. The advantages of the simulation is that we can easily interpolate/calculate efficiency values for any torque-velocity point, as well as make the efficiency map more granular by decreasing the spacing between the uniform points. Additionally, with the simulation, the efficiency in all motor operation quadrants can be calculated. It shows that quadrants I & III (forward operation) are symmetrical, and II & IV (regeneration) are symmetrical, and thus it should not be an issue to only test quadrant I & IV during physical modeling. However, there are some very noticeable differences, mainly the 0/-1 Nm torque line and the saturation limit seen in Figure 29 b). This is due to the simulation not taking into account physical real world limitations of an actuator, such as the saturation limit of a motor.

3.3.3 Biomechanics Data

Using the 10 subjects biomechanics for walking at different inclines and speed, the electrical power, mechanical power and actuator efficiency through their gait cycle can be calculated. Figure 30 shows the actuator efficiency, mechanical power, electrical power for subject 3 walking at a speed of 1 m/s on an incline of 0º.

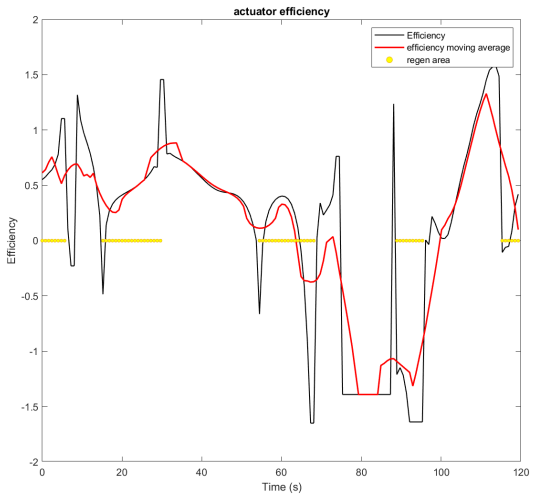
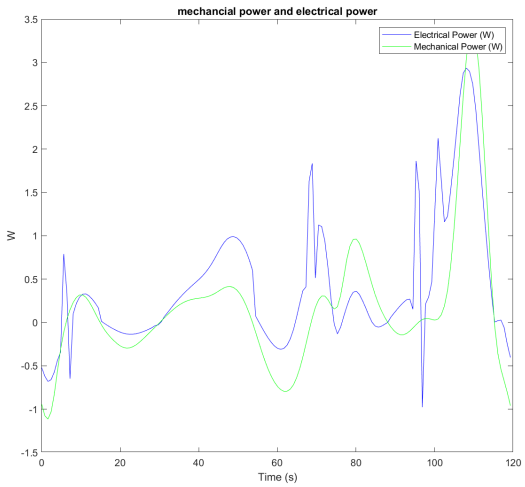
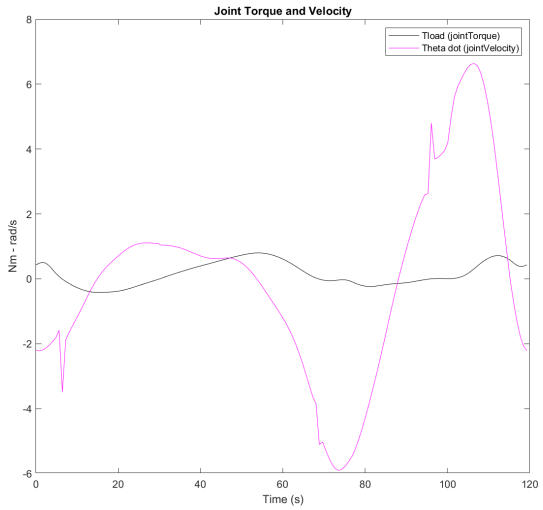


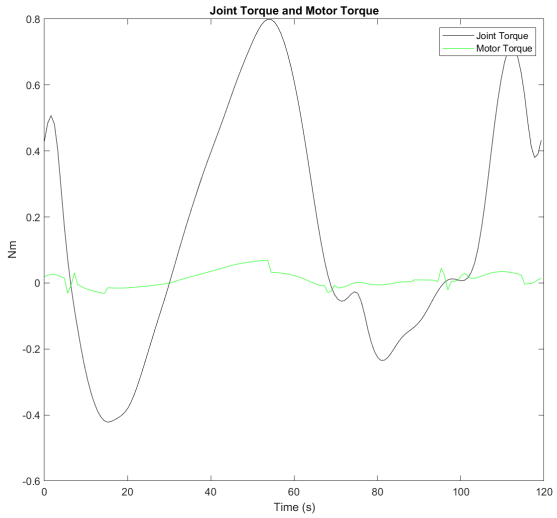
Figure 30 a) Actuator Efficiency During a Human Gait Cycle



b) Mechanical and Electrical Power



c) Inputs to the Simulation



d) Output Shaft Torque (Joint Torque) vs Motor Torque

Using the electrical power, the battery capacity and the area where regeneration occurs, the amount of steps that can be taken with the available battery capacity is calculated. Using a similar method as [1], the total number of steps is calculated for both regeneration and without regeneration, to compare the 2 values and obtain the %increase with regeneration active. The formula to calculate steps is found below:

$$T_{steps} = \frac{Battery\ Capacity}{Energy\ Consumed - Energy\ Regenerated} \div 2$$

Any points of negative efficiency are considered as power consumption. Table 5 below shows the percentage difference between different inclines for different subjects walking at 1 m/s.

Table 5: Percentage increase in total number of steps at different inclines for different subjects

Subjects	Percentage Increase (%)				
	-10°	-5°	0°	5°	10°
1	260.86	589.14	59.74	154.79	16.94
2	172.54	53.21	28.71	29.69	20.94
3	0	85.63	46.57	26.58	17.58
4	108.19	0	76.80	0	0
5	0	85.91	0	0	0
6	22.91	44.30	52.50	42.71	47.35
7	208.60	103.44	40.79	29.71	62.92
8	230.56	75.91	97.90	478.9	271.61
9	295.65	0	22.41	16.32	174.75
10	165.90	66.14	49.94	63.38	21.29

Ignoring the Zero values and the outliers, such as subject's 1 +589% at -5° incline or subject's 8 +478.9% at 5^0 incline, an average can be calculated, resulting in Table 6.

Table 6: Average of all subjects, with outliers and 0 removed, and compared to [1]

Incline	-10°	-5°	0°	5°	10°
Simulation % Increase	205	73	52	34	30
[1] % increase	74	41	14	6	4

Though our calculation for the percentage increase follows a similar pattern to [1], where the amount of regeneration decreases with increasing inclines, our values are vastly different from [1]. Firstly, our %increase is much higher, but this is due to our simulation only taking into account the knee joint, which has the highest regeneration potential [1], while [1] considers all joints including the hip and the ankle. Additionally, for the same battery capacity 967 680 J, our simulation got a much lower total number of steps with our steps being an order of magnitude lower, in the 10^3 range while [1] was in the 10^4 range. This could either be due to our model being more accurate and making less assumptions, or issues in our code/data processing technique. To be certain, a more in depth investigation of the data and model is required. Lastly, the percentage increase between each subject was vastly different, introducing large error bars. We were unable to investigate whether this is due to some anomaly in the data or anomaly in the code.

Though these results look promising, they should be read with reservations. There are many areas that require refinement and validation. The main points to keep in mind while reading these are:

1. The Simulation model was not validated with ground truth data, the benchtop
2. Our motor and actuator constants are from [2], and was not obtained directly from our benchtop
3. We are assuming the actuator would follow the human gait cycle perfectly, but this is likely impossible since it is both hard and not the most optimized way of using regeneration [8]

4. From Figure 30 a), we are observing negative efficiencies in the forward motoring quadrant, which is unexpected and not observed in any of the benchtop simulation efficiency maps. This is of high concern and would require us to look deeper into both the equations and the data.

4.0 Future Work

This section highlights next steps and parts of the project which future students should work on and consider.

4.1 Mathematical Modelling

The ultimate goal of this area of research is to regenerate energy to optimize battery life in wearable robotics. As mentioned previously, this project was cut short by the mechanical and electrical issues which could not be solved in time with the resources available. Referencing Appendix A, the team got results for the “Physical Modelling" and “Testing” sections but was unable to dive into “Mathematical Modeling”. The idea was to perform system identification on the dynamometer setup, a process in which data is used to develop a model of a dynamic system. In Section 3.0, the team captured the essential underlying dynamics, along with the relevant parameters of the actuator’s energy efficiency when loaded as a pendulum. From there, actuator data from experimental procedures outlined in this report can be used to complete the mathematical model. Once a model structure has been identified, the free parameters can be optimized by minimizing a cost function. This process is generalized in Figure 31; it should be noted that Matlab is a powerful tool to perform system identification. The purpose of this model is to predict the actuator’s behavior outside of its operating range; currently, the 3D printed actuators are unable to reach the higher torque-speed pairs from the biomechanics data. Outside the scope of this project, future work should include the development of a similar model for more powerful actuators. From there, controllers may be developed, validated, and implemented to regenerate energy and extend battery life.

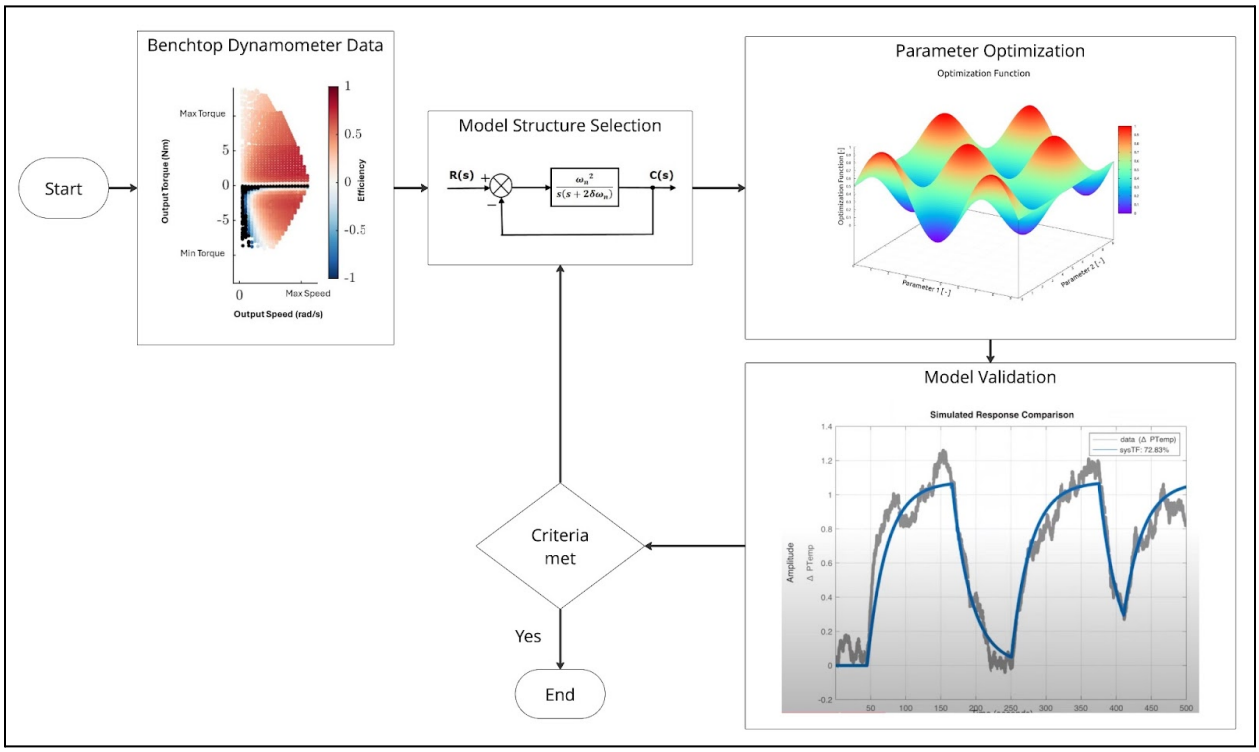


Figure 31: System Identification Process and Example [9]

4.2 Mechanical system

There are a few actuator improvements and actions that can be taken to mitigate the likelihood of similar mechanical problems occurring in the future. One of which being to utilize heat-set inserts for removability of the 3D-printed components in order to stay true to the highly customizable aspect of the design that Urs et al. had envisioned. Another being the utilization of a torque gauge when ensuring that the housing is securely fastened. Once the proper minimum N·m torque can be experimentally found, this will limit the chances of having a housing improperly fastened before undergoing significant vibratory motor rotational movement. Finally, future groups who need to make component modifications can make sure to cross-reference the BOM that our group created to ensure that, for example, the motor coils will not snap due to having planet carrier fasteners missing from the assembly.

4.3 Electrical System

As mentioned in Section 2.3, the Power Distribution Board should be replaced if the budget allows. Currently, there is a makeshift solution in place that wires up the switch JST.

One important update that would improve the usability and longevity of the system is the inclusion of a Battery Management System. Usually, these are installed at the terminals of a LiPo, and connected to the balance port of the LiPo. These systems usually have numerous safety and preservation features, including overcharge and under-charge protection, overcurrent protection, cell balancing, and more. In a real powered prosthetic with regeneration, a BMS system would be invaluable as sudden switches between charging and discharging can lead to high cell voltage imbalances, reducing the lifespan of the LiPo battery. Another improvement that would reduce the likelihood of failures are to replace outdated components to newer offerings. Of the benchtop components, only the motor controllers themselves are the most updated models; the Pi3 Hat and PDB are both 2 versions old. Newer versions of these aforementioned products have robustness and reliability improvements which can prove invaluable to the smooth running of the Benchtop as a whole.

4.4 Software System

Some difficult issues arose here relating to interfacing with Moteus in a multithreaded program. In the current state of the program, the sensors, the logger, and even the arbitrator FSM logic work as expected. The main issue is that the torque controller does not track torque properly. For example, if motor 1 starts in velocity control, and then motor 2 is commanded a torque of 1 while SetStop is called on motor 1, the torque controller will maintain the same velocity as transmitted by motor 1 earlier. After the torque controller is fixed, the condition for transitioning from torque ramp up to test point hold should be reviewed. We have observed large fluctuations on Tview when running torque control in Figure 32, which is why the steady state tolerance was set to quite high, but perhaps this is an issue that can be fixed as well. Potential causes of this include imbalances in the mechanical component which causes the actuator to turn unevenly.

Unit tests were written for most components except for the torque/velocity controllers and the arbitrator. Using Moteus with GTest posed significant challenges. There are many magic numbers in the arbitrator code, such as the steady state window, which should be finetuned by hand. Controller parameters such as gains should also be scrutinized.

Although scripts for the GUI and efficiency graph generation are written and tested individually, they have not been tested end-to-end with actual data from the motors collected by the C++ code. There may be some minor misalignment issues between what the scripts expect and what the actual data looks like.

5.0 Conclusion

To summarize, despite encountering several challenges during the course of this project, the team has demonstrated the ability to apply engineering principles effectively in navigating through various iterations to improve our approach. Our initial aim was to address the problem of limited energy capacity in wearable robotics by developing a control system for energy regeneration based on a predictive model. However, the team faced setbacks in achieving tangible results due to unforeseen issues with the actuators, particularly the sheared planetary gear, stripped housing threading, and the snapped motor coils. Despite these challenges, the team successfully conducted benchtop testing and simulations to characterize the actuator's efficiency and validate our model. Moving forward, we will refine the system identification process, improve actuator design, and explore the potential for energy regeneration beyond the actuator's operating bounds. Overall, while we did not achieve all of our intended results, this project has still provided valuable insights and laid the groundwork for future research in the field of wearable robotics and in energy regeneration control.

6.0 References

[1] B. Laschowski, K. A. Inkol, A. Mihailidis, and J. McPhee, Simulation of energy regeneration in human locomotion for efficient exoskeleton actuation, 2022. doi:10.1101/2022.06.13.495983

[2] K. Urs, C. E. Adu, E. J. Rouse, and T. Y. Moore, “Design and characterization of 3D printed, open-source actuators for legged locomotion,” 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Dec. 2022. doi:10.1109/iros47612.2022.9981940

[3] T. Verstraten, R. Furnémont, G. Mathijssen, B. Vanderborght and D. Lefeber, "Energy Consumption of Geared DC Motors in Dynamic Applications: Comparing Modeling Approaches," in IEEE Robotics and Automation Letters, vol. 1, no. 1, pp. 524-530, Jan. 2016, doi: 10.1109/LRA.2016.2517820.

[4] T. Verstraten, G. Mathijssen, R. Furnémont, B. Vanderborght, D. Lefeber, “Modeling and design of geared DC motors for energy efficiency: Comparison between theory and experiments”,Mechatronics, Volume 30, 2015, Pages 198-213, ISSN 0957-4158, https://doi.oq1rg/10.1016/j.mechatronics.2015.07.004.

[5] “Micro carbon fiber filled nylon that forms the foundation of Markforged composite parts,” Markforged [Online]. Available: https://markforged.com/materials/plastics/onyx

[6] B. Laschowski, “Benchtop Testing.” Bionics Lab, University of Toronto, Toronto, Ontario, Canada, 2023

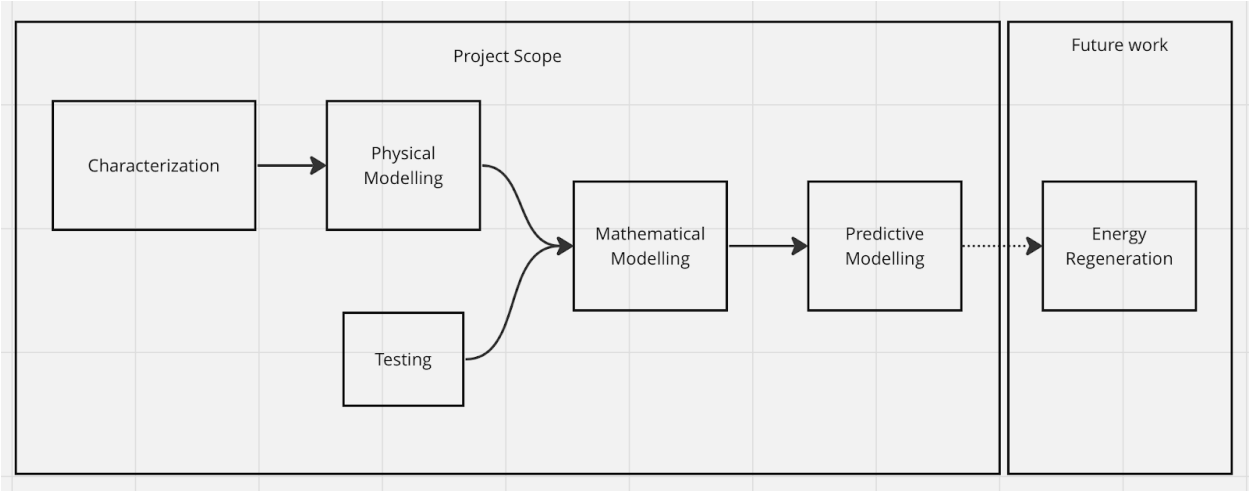
[7] R. Gregg, “Lower-limb kinematics and kinetics during continuously varying human locomotion,” figshare, https://doi.org/10.6084/m9.figshare.c.5175254.v1

[8] G. Khademi, H. Richter, and D. Simon, “Multi-objective optimization of tracking/impedance control for a prosthetic leg with energy regeneration,” 2016 IEEE 55th Conference on Decision and Control (CDC), Dec. 2016. doi:10.1109/cdc.2016.7799085

[9] B. Douglas, “Linear System identification | system identification, part 2,” YouTube, https://www.youtube.com/watch?v=qC_C04SEV1E&t=560s

7.0 Appendix

Appendix A: Problem resolution plan for optimizing robot battery life



Appendix B: Legend of Variable Symbols, Meanings, and Values

Variable	Symbol	Value
Motor Inertia	J_m	90.1 g/cm^2
Gear inertia	J_g	0.219 kg/m^2
Gear Ratio	n	15:1
Gear Efficiency	η_a	0.72
Pendulum Length	l	0.5 m
Gear Damping	v	0.03
Motor Damping	v_m	$0.004\frac{Nm}{rpm}$
Motor Torque Constant	k_t	105 mNm/A
Motor Speed Constant	k_b	0.094 rpm/V
Motor Resistance	R	1.42 Ohm
Motor Inductance	L	1.5 mH
Gravitational acceleration	g	-9.81 $\frac{m}{s^2}$
Mass	M	0.536 kg