Evaluation of a Decentralized Stance-Leg Controller on a Powered Robotic Leg

Yin Zhong

CMU-RI-TR-16-15

April 2016

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Hartmut Geyer (Chair)
Koushil Sreenath
Jiaji Zhou (Student Member)

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Keywords: Robot,Prosthesis,Bipedal,Locomotion,Neuromuscular,Control,Simulation

For Troy, my sincere friend.

Abstract

Centralized controlling approaches, such as full-body trajectory planning and tracking, are prevalent in locomotion control of humanoid robots. However, they cannot be applied to powered prosthetic devices as a dynamic model is not available for the human in the loop, and it is difficult to estimate state of the human. One alternative is based on replay of motion patterns captured from able-bodied human; however, their performance is unknown under large disturbances. In contrast, heuristics-based controllers encode motion patterns in feedback control laws and/or state machines. One such approach called neuromuscular control stems from studies on animal and human locomotion behavior, which has potential to enable diverse and robust locomotion behaviors in both humanoids and prosthetic devices. As a part of ongoing effort to evaluate this approach on hardware, this thesis presents work on transferring a neuromuscular stance-leg controller to Robotic Neuromuscular Leg 3 (RNL3), a powered segmented leg hardware platform. The controller has been shown in simulation of an idealized 2-segment leg to produce a stable bouncing gait. It is implemented and tested in a physical simulation model of RNL3, then transferred and tested on actual hardware. The controller is capable of maintaining a stable in-place bouncing gait in simulation. However, the robotic leg suffered from mechanical failures during experiments; as a result the controller remains to be verified on hardware.

Acknowledgments

Firstly I would like to thank my academic advisor Hartmut Geyer, who has relentlessly mentored and motivated me during my whole course of study. He patiently guided me through confusions and frustrations encountered in this work, and gently brought me into the world of academic research. I would also like to thank my committee members Koushil Sreenath and Jiaji Zhou for their time and support.

Thank you to Alexander Schepelmann, who had been the previous warden of the Robotic Neuromuscular Leg project. He devoted tremendous effort on all three generations of the robot, including the one serving as the subject of this work, from design to manufacturing and even maintenance, even after he graduated from PhD program. I could not have a functional robot to test on without help from Alex and other students in the lab: Nitish Thatte, William Martin, Akshara Rai. They generously lent me their hand when hardware work became too hard for me alone. Also, their sense of humor gave me emotional relief. Additional thanks to Albert Wu and Seungmoon Song who provided valuable feedback on simulation and control.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents work on transfering a controller that produces spring-like stance behavior in segmented legs based on a neuromuscular reflex model [8] onto a powered robotic leg.

1.1 Motivation

Legged locomotion is a powerful mobility tool for humans and animals alike. From observation of legged locomotion in nature, human have built both standalone walking and running machines, as well as passive and active powered prosthetic devices. A prevalent approach used in humanoid robots for locomotion control is full-body trajectory planning and tracking, in which planned footsteps are first converted to trajectories of the whole body, then tracked at all joints. State-of-the-art controllers following this approach, such as one running on HUBO [17], which recently won The DARPA Robotics Challenge 2015, center around the the concept of "Zero Moment Point" (ZMP) [34], which aligns the cen-

ter of mass (CoM) of the whole robot with the center of pressure (CoP) of its foot, so that ground reaction force (GRF) does not produce moment on the robot during stance phases. However, in order to convert CoM trajectories into joint space, full knowledge of the system is required, including accurate knowledge of its dynamic model and estimation of all states; full control over all joints is also required. As a result, it cannot be applied to powered prosthetic devices, as the human is a part of the system, whose state is difficult to accurately estimate and joints not under direct control. Instead, control strategies based on replay of motion patterns captured from able-bodied human were developed for prosthetic and orthotic applications, such as [2], which have been shown to work well on active prosthesis devices. Motion patterns are not limited to joint position [1], but also joint impedance [27], or even locus of CoP in a particular frame [10]. Such strategies were applied to active powered transfemoral prostheses, enabling wearers to walk over level and sloped ground, even assist in other common tasks such as standing up and sitting down [33]. However, their performance is unknown when subjected to large disturbances such as uneven terrain or tripping over obstacles.

In contrast to these centralized approaches, heuristics-based controllers encode the desired motion patterns in feedback control laws and/or finite state machines. One particular approach stems from studies on animal and human locomotion behavior, which revealed that low-level spinal reflex circuits including central pattern generators (CPGs) are vital to leg locomotion even with no input from brain [15, 16]. Inspired by these findings, a series of decentralized, heuristic controllers have been developed. A controller consisting of a single Hill-type muscle model [14] stimulated by positive force feedback has been shown to produce a stable bouncing gait in simulation of an ideal 2-segment leg [8]. The idea was extended to apply on a 7-segment model consisting of one trunk and two 3-segment legs [6]. Controller for it has 7 Hill-type muscle models per leg, associated neuromuscular reflex feedback network, and a simple state machine. It generates a stable walking gait and adapts

to both small ground disturbances ($< \pm 4\,\text{cm}$) and slopes ($< \pm 4\,\%$) without any prior knowledge or parameter tuning. More recently, an alternative controller for the swing phase was developed to achieve accurate swing leg placement, which helps maintain gait stability [23]. The controller was first conceived using only heuristics and simple feedback laws [3, 22], then reformulated into neuromuscular reflexes [4]. These neuromuscular controllers have since been further extended to demonstrate even diverse behavior, such as running [26], climbing stairs, turning, and stepping over obstacles [25].

While these controllers were initially developed in simulation, verification work has been done by transferring them onto both standalone robotic systems and prosthetic devices. A controller containing a single muscle model[1] and corresponding reflexes was developed for a powered ankle-foot prosthesis [5], which was then tested successfully on a transtibial amputee. A powered transfemoral prosthesis prototype is under development to evaluate neuromuscular control [31]. It currently has a powered knee joint and a passive spring-loaded ankle joint. The controller implemented is derived from [6, 22] with ankle and hip torques removed. It was successfully tested first in simulation with a simulated human model, then on an able-bodied subject through adapters (knee crutch and lift shoe). In order to further study the behavior and performance of the controllers as well as the gaits they produce, fully robotic platforms have also been developed. They are segmented legs powered by series-elastic actuators, details of which described in chapter 3. They are designed to enable rigorous and repeatable experiments on neuromuscular control strategies as well as comparing them to other controllers. Swing-leg controllers described in [3, 4] have been successfully transferred and tested on these platforms [20], which achieved accurate and human-like swing-leg placement even under disturbances in the form of unexpected obstacles. The goal of the platform is to achieve full walking behavior. However, it currently lacks controller for the stance phase. Such motivates research presented in this thesis.

[1] Ankle extensor, not the knee extensor in the bouncing controller

1.2 Hypotheses and Thesis Outline

This thesis investigates whether a neuromuscular stance control strategy can be applied to powered segmented legs by transferring the ideal neuromuscular bouncing controller on actual hardware. Specifically, the following hypotheses are discussed:

1. The controller is capable of generating compliant stance leg behavior.
2. The controller is capable of maintaining a stable in-place bouncing gait.

Chapter 2 introduces state-of-the-art approaches in prosthesis control and the main approach of this research. Development of a dynamically scaled powered robotic leg that has the capabilities needed for evaluating stance-leg controllers is described in chapter 3. Preliminary experiments on this hardware are also discussed. Chapter 4 presents development of a simulation of the physical system, on which the neuromuscular controller is tuned, and tested. Results from simulated experiments are analyzed. Finally, chapter 5 presents discussion and future work of this research.

Chapter 2

Background

2.1 Control Strategies for Prosthetic Devices

2.1.1 Impedance Control

Material in this section contains summary of the following: [19, 27–29, 32, 33]

The state-of-the-art control strategy for prosthetic devices is impedance control, originally developed at Vanderbilt University for their powered transfemoral knee-ankle prosthesis. It is based on the idea that torque at each joint in a particular gait can be fitted piecewise as a simple function of angle and velocity of the joint:

$$\tau = k_1 (\theta - \theta_e) + k_2 (\theta - \theta_e)^2 + b\dot{\theta} \tag{2.1}$$

which is equivalent to a virtual (non-linear) spring-damper across the joint. Each set of parameters (k_1, k_2, θ_e, b for each joint) then encodes the behavior of one segment in gait.

Control over the whole gait period requires a number of parameter sets. In the original implementation of impedance walking controller, each of the stance and swing phases are sub-divided into 2 "modes" (parameter sets):

1. Early stance / Stance flexion: bear weight
2. Late stance / Pre-swing: ankle push-off until toe-off
3. Early swing / Swing flexion: clear ground and forward swing
4. Late swing / Swing extension: extend knee until heel-strike

These "modes" are then represented as a finite state machine which selects the active parameter set for the underlying impedance controller. States are switched using simple heuristic criterions based on input from sensors mounted on the prosthesis. This controller was tested on prototype hardware and able-bodied human subject through usage of prosthetic testing adapters. It produced near-normal level ground walking gait on a treadmill. Later, the state machine was expanded to include more modes for both level walking and standing. It was implemented on a standalone version of the hardware and tested on a unilateral transfemoral amputee subject. The prosthesis was tuned on a treadmill to the subject and then tested over-ground. This series of controllers have since been expanded to include more behavior such as sit-stand transition and upslope walking.

A potential problem of impedance control is that, if the system of amputee and prosthesis encounters disturbance such as obstacles, unless detected by the state machine or higher level control, parameter set of the impedance controller remain unchanged, which might lead to instability or even falls. On the other hand, while a machine-learning-based stumble detector and classifier were developed for impedance controllers, it has not been shown how to model the necessary recovery actions in order to recover from detected disturbance. Also, as more robust behaviors are desired, more parameter sets and corresponding state

transitions need to be designed and tuned, which would greatly complicate the design of the whole controller.

2.1.2 Virtual Constraint Control

Material in this section contains summary of the following: [9–11]

A control strategy for walking stance called Virtual Constraint Control was recently proposed as an alternative to impedance controllers. It addresses the difficulty in reliably tuning existing control strategies. Inspired by output linearization techniques in fully robotic systems, its core idea is based on softly enforcing a set of holonomic constraints on the powered open kinematic chain through actuator torques. These constraints are derived from observation on the progression of the center of pressure (CoP) during human walking stance: It moves monotonically from heel to toe, and its trajectory is invariant over subjects and walking conditions in two frames, both with origin placed at ankle — one rigidly attached to the shank, the other with one axis passing through hip (i.e. attached to the virtual leg). A holonomic constraint function $h_s(q)$ is defined to be deviation of actual CoP in the prosthesis from the prescribed trajectory. The goal of the controller is then to drive this constraint function to zero, which effectively synchronizes knee and ankle movements to current CoP position, which in turn acts as the single phase variable of the system. Output linearization was employed to create a control law that drives the constraint function to zero with only feedback provided by sensors on the prosthesis, namely joint angles and force sensor readings.

The controller was first tested in simulation with a simple ideal bipedal model to produce downhill passive walking gait; the bipedal model was then improved so that a level-ground walking gait could be produced. This controller has been transferred on both the Vanderbilt

prosthetic leg and an alternative prosthetic hardware developed at UT Dallas, and tested on both an able-bodied human subject and an amputee subject to produce level-ground treadmill walking gait up to 2 miles per hour.

While this approach alleviates the need to tune multiple sets of parameters on individual amputees at least for the stance phase, it has not been demonstrated to produce gaits other than level-ground walking, such as walking on slopes. It is not known either whether or not the CoP trajectory during human walking stance remains invariant when ground slope changes, or when the amputee-prosthesis system encounters disturbances.

2.2 Approach

Neuromuscular control strategy was inspired by studies on animal and human legged locomotion behavior. It has shown the potential of enabling robust and human-like gaits in both prosthetic devices and fully robotic systems. Instead of reasoning directly with motion patterns in configuration space, neuromuscular control attempts to address fundamental functions that are required for a leg, regardless of biological or robotic, to perform basic locomotion tasks. In general, a neuromuscular controller consists of two distinct parts: a virtual neuromuscular model, and a reflex layer. The virtual neuromuscular model contains multiple Hill-type muscle-tendon unit (MTU) models attached to a virtual skeleton. The skeleton maps the robot configuration to MTU inputs, and MTU outputs to robot actuation commands. This part is common to all neuromuscular controllers. The reflex layer contains simple feedback laws that encode each desired leg function, similar to spinal reflexes in human legs. Since different locomotion tasks require different functions, neuromuscular controllers for each task has a distinct reflex layer. The following sections elaborate the details of each component listed above.

2.2.1 Hill-type Muscle-Tendon Unit Model

Material in this section contains summary of [6, 8].

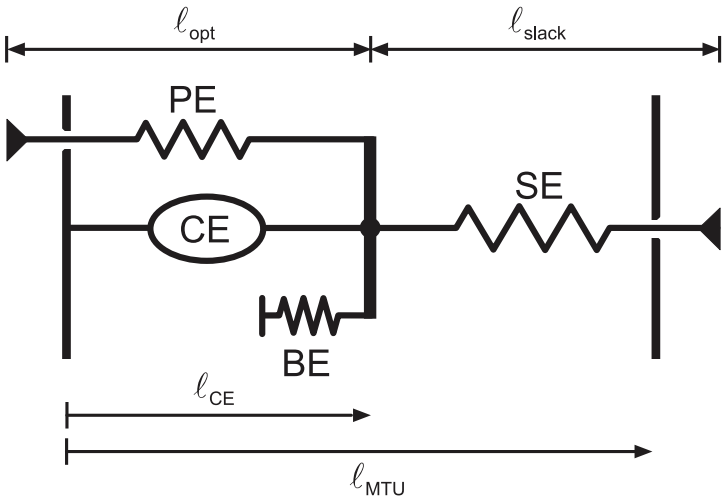


Figure 2.1: Schematic of Hill-type MTU model

Hill-type muscle-tendon unit model approximates the dynamics of human leg muscles and tendons. It is an 1D mechanism consisting of an active contractile element (CE) and three passive springs (SE, PE, BE). Input to the model is total length of the MTU l_{MTU} and a normalized muscle stimulation signal $0 \leq S \leq 1$. Its main output is the total force F_{MTC}.

The contractile element (CE) models force production in the muscle by relating its contraction force to the length l_{CE} of CE and its derivative (velocity) v_{CE} as:

$$f_{\mathrm{CE}} = A \cdot f_l\left(l_{\mathrm{CE}}\right) f_v\left(v_{\mathrm{CE}}\right) \tag{2.2}$$

$$F_{\mathrm{CE}} = F_{\max} f_{\mathrm{CE}} \tag{2.3}$$

where f_l and f_v denotes the force-length and force-velocity relationships respectively, defined in [8]; A denotes muscle activation, which is related to the stimulation input through

"excitation-contraction coupling" modeled as a linear low-pass filter:

$$\tau \dot{A} = S - A \tag{2.4}$$

$$A|_{t=0} = 0 \tag{2.5}$$

namely the transfer function $\frac{1}{1+\tau s}$.

The series elasticity (SE) element is a non-linear spring modeled as:

$$f_{\mathrm{SE}} = \begin{cases} \left(\varepsilon/\varepsilon_{\mathrm{ref}}\right)^2 & \varepsilon > 0 \\ 0 & \varepsilon \leq 0 \end{cases} \tag{2.6}$$

$$F_{\mathrm{SE}} = F_{\max} f_{\mathrm{SE}} \tag{2.7}$$

where $\varepsilon = (l_{\mathrm{SE}}/l_{\mathrm{slack}}) - 1$ is the strain of SE, and the elasticity is defined through reference strain $\varepsilon_{\mathrm{ref}}$ under which the SE produces maximum force $F_{\mathrm{SE}} = F_{\max}$.

While there are three springs in the mechanism, the parallel elasticity (PE) and buffer elasticity (BE) elements are "safeguards" that are only active outside main operating range of the mechanism. When PE and BE are inactive, the CE and SE are effectively connected in series and therefore their forces are equal. This relationship allows solving the internal dynamics of the MTU. By substituting $f_{\mathrm{CE}} = f_{\mathrm{SE}} = f_{\mathrm{MTU}}$ and $v_{\mathrm{CE}} = \dot{l}_{\mathrm{CE}}$ into CE and SE dynamics[1]:

$$A \cdot f_l\left(l_{\mathrm{CE}}\right) f_v\left(\dot{l}_{\mathrm{CE}}\right) = \left(\varepsilon/\varepsilon_{\mathrm{ref}}\right)^2 \tag{2.8}$$

f_v is an invertible function, therefore[2]:

$$\dot{l}_{\mathrm{CE}} = f_v^{-1}\left(\frac{\left(\varepsilon/\varepsilon_{\mathrm{ref}}\right)^2}{A \cdot f_l\left(l_{\mathrm{CE}}\right)}\right) \tag{2.9}$$

[1] Assuming SE is active; the inactive case is trivial as no force is produced.

This ODE can then be integrated to determine force output of MTU. Also, Also, normalized variables in the system are provided as feedback to the neuromuscular reflex layer:

$$\hat{f} = F_{\text{MTU}}/F_{\max} \tag{2.10}$$

$$\hat{l} = l_{\text{CE}}/l_{\text{opt}} \tag{2.11}$$

$$\hat{v} = v_{\text{CE}}/v_{\max} \tag{2.12}$$

2.2.2 Virtual Skeleton

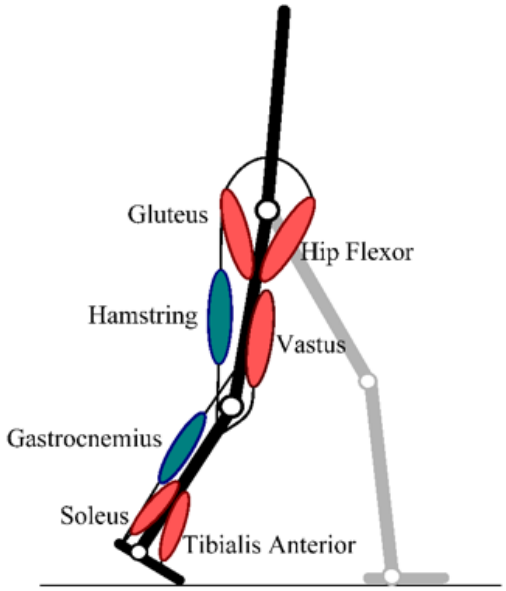


Figure 2.2: The 2D neuromuscular model

The virtual skeleton is the interface between the MTUs and the actual robot hardware. Fig. 2.2 shows the attachment of all MTUs (listed in Table 2.1) on the virtual skeleton. The joint angles of the skeleton are the same as that of the robot. Geometry of the skeleton determines the instantaneous moment arm r_{MTU} and length l_{MTU} for each MTU. Notice

[2]Here division-by-zero can be prevented with a small lower bound on A and activation of PE and BE.

Table 2.1: Muscle groups in 2D neuromuscular model and their effects on joints (Ext. for extensor, Flx. for flexor)

Name	Abbr.	Hip	Knee	Ankle
Gluteus	GLU	Ext.	/	/
Hip Flexor	HFL	Flx.	/	/
Hamstring	HAM	Ext.	Flx.	/
Rectus Femoris	RF	Flx.	Ext.	/
Vastus	VAS	/	Ext.	/
Biceps Femoris Short Head	BFsH	/	Flx.	/
Gastrocnemius	GAS	/	Flx.	Ext.
Soleus	SOL	/	/	Ext.
Tibialis Anterior	TA	/	/	Flx.

that r and l are related through conservation of energy:

$$\tau d\phi = Fdl \tag{2.13}$$

$$\rho F r(\phi) d\phi = Fdl \tag{2.14}$$

where ρ is a constant that adjusts for muscle pennation angles (considered as a length scaling). Solving for l:

$$l(\phi) = l_{\text{ref}} + \rho \int_{\phi_{\text{ref}}}^{\phi} r(\phi) d\phi \tag{2.15}$$

where l_{ref} and ϕ_{ref} specify the muscle length at a reference configuration. Effectively, only the moment arm for each MTU needs to be specified in terms of joint angle. This geometry mostly follows [6] with a few exceptions. Firstly, all parameters in the virtual neuromuscular model are 1/2 dynamically scaled in agreement with hardware (section 3). Muscle attachments on hip are considered to have constant moment arm. Those on knee and ankle are considered to have variable moment arm. One such relationship was specified in [6], which is an approximation of human anatomy. However, while it was suitable for walking, the knee joint could potentially be flexed more than 90° during in-place bouncing[3]

, and the approximation is no longer appropriate. Instead, human data from [12] was fitted as a 6th order polynomial, which covers data points from 80° through 180°. This polynomial fit is shown in Fig. 2.3. An additional advantage of using polynomial is ease of integration (for muscle length) and numerical calculation.

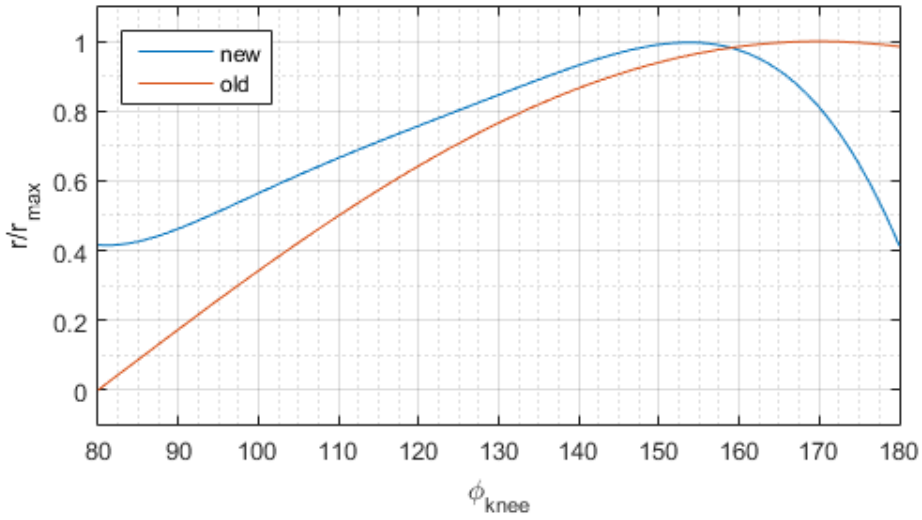


Figure 2.3: Comparison of new and old variable moment arm relationships

In general, mapping of the MTU forces to robot actuation involves calculating the equivalent torque of MTUs at joints of the virtual skeleton, then arrange actuation to produce the same torque on corresponding joints on the robot. Details of this mapping to the hardware testbed is described in section 4.1.3.

2.2.3 Stance Reflex Layer

The reflex layer as a whole receives *time-delayed* feedback signals from MTUs and joint angles from encoders. It determines stimulation for each MTU according to feedback laws that encode locomotion functions. For walking and running stance, the most important

[3]In the original bouncing paper [8], the moment arm was simplified to be constant, so that this problem was not exposed.

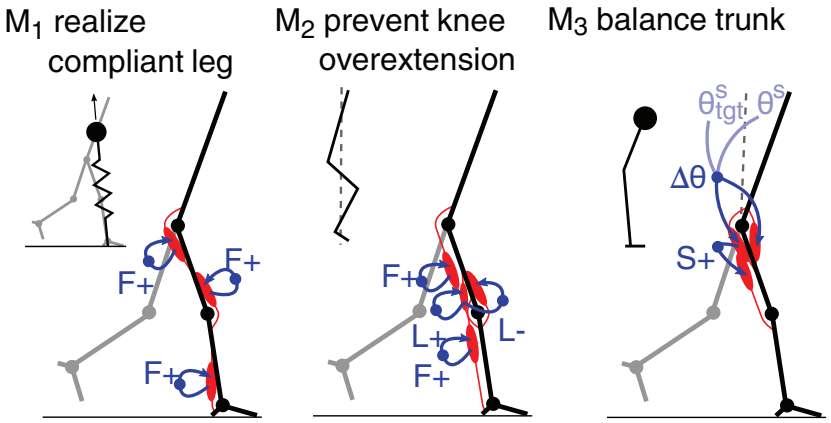


Figure 2.4: Schematic of Hill-type MTU model

functions include realizing compliant leg behavior [7] and maintaining leg stability [24]. Each function is implemented in the stance reflex layer as a "module", as described in Fig. 2.4 from [25].

M1 is responsible for compliant leg behavior. It consists of positive linear force self-feedbacks on extensor muscles:

$$\text{GLU} \xrightarrow{F+} \text{GLU} \qquad S^{\text{GLU,M1}} = G^{\text{GLU}}_{\text{GLU}} \left[\hat{f}^{\text{GLU}}\right]_+ \tag{2.16}$$

$$\text{VAS} \xrightarrow{F+} \text{VAS} \qquad S^{\text{VAS,M1}} = G^{\text{VAS}}_{\text{VAS}} \left[\hat{f}^{\text{VAS}}\right]_+ \tag{2.17}$$

$$\text{SOL} \xrightarrow{F+} \text{SOL} \qquad S^{\text{SOL,M1}} = G^{\text{SOL}}_{\text{SOL}} \left[\hat{f}^{\text{SOL}}\right]_+ \tag{2.18}$$

M2 addresses potential instability in a multi-segmented leg caused by over-extension. It mostly involves knee muscles:

$$\text{HAM} \xrightarrow{F+} \text{HAM} \qquad S^{\text{HAM,M2}} = G^{\text{HAM}}_{\text{HAM}} \left[\hat{f}^{\text{HAM}}\right]_+ \tag{2.19}$$

$$\text{GAS} \xrightarrow{F+} \text{GAS} \qquad S^{\text{GAS,M2}} = G^{\text{GAS}}_{\text{GAS}} \left[\hat{f}^{\text{GAS}}\right]_+ \tag{2.20}$$

$$\text{BFsH} \xrightarrow{L+} \text{BFsH} \qquad S^{\text{BFsH,M2}} = G^{\text{BFsH}}_{\text{BFsH}} \left[\Delta\hat{l}^{\text{BFsH}}\right]_+ \tag{2.21}$$

$$\text{BFsH} \xrightarrow{L-} \text{VAS} \qquad S^{\text{VAS,M2,1}} = -G^{\text{VAS}}_{\text{BFsH}} \left[\Delta \hat{l}^{\text{BFsH}}\right]_+ \tag{2.22}$$

$$\text{knee} \xrightarrow{\theta-} \text{VAS} \qquad S^{\text{VAS,M2,2}} = -G^{\text{VAS}}_{\text{knee}} \left[\theta^{\text{knee}} - \theta^{\text{knee}}_{\text{off}}\right]_+ \tag{2.23}$$

where $\Delta\hat{l} = \hat{l} - \hat{l}_{\text{off}}$ compares normalized length feedback to a constant offset. This offset is set at where the knee enters over-extension.

M3 attempts to maintain trunk balance by regulating trunk angle in world frame to zero (or any given reference). It is different from other feedbacks in that it consists of a traditional PD control loop over trunk angle error; the output of the PD loop is fed to either the hip extensors (GLU and HAM) or hip flexors (HFL and RF) depending on sign.

Besides M1 through M3, there are other modules encoding more functions for the complete 3D ideal bipedal model in [25]; however, they are not relevant to the task of in-place bouncing stance, therefore omitted from discussion.

The final stimulation signal for each muscle is the sum of contributions from all 3 modules, plus a constant bias S_0 named "pre-stimulation", then hard clipped to the interval $[S_{\min}, 1]$, where $S_{\min}$ is a small constant (e.g. 0.01) to avoid division-by-zero in muscle model calculations.

2.2.4 Controller Modification due to Hardware Limitation

The hardware platform as it is currently implemented imposes a certain limitations (see section 3.1) and does not completely match the neuromuscular model described above. The following changes to the controller are made to adapt to these limitations:

1. Reflexes involving ankle actuators (SOL and TA) are disabled (because point foot

replaces ankle-foot)

2. GAS reflex (only one in M2) is disabled (because GAS SEA acts as BFsH in current hardware)
3. M3 is entirely disabled (because trunk angle is locked)

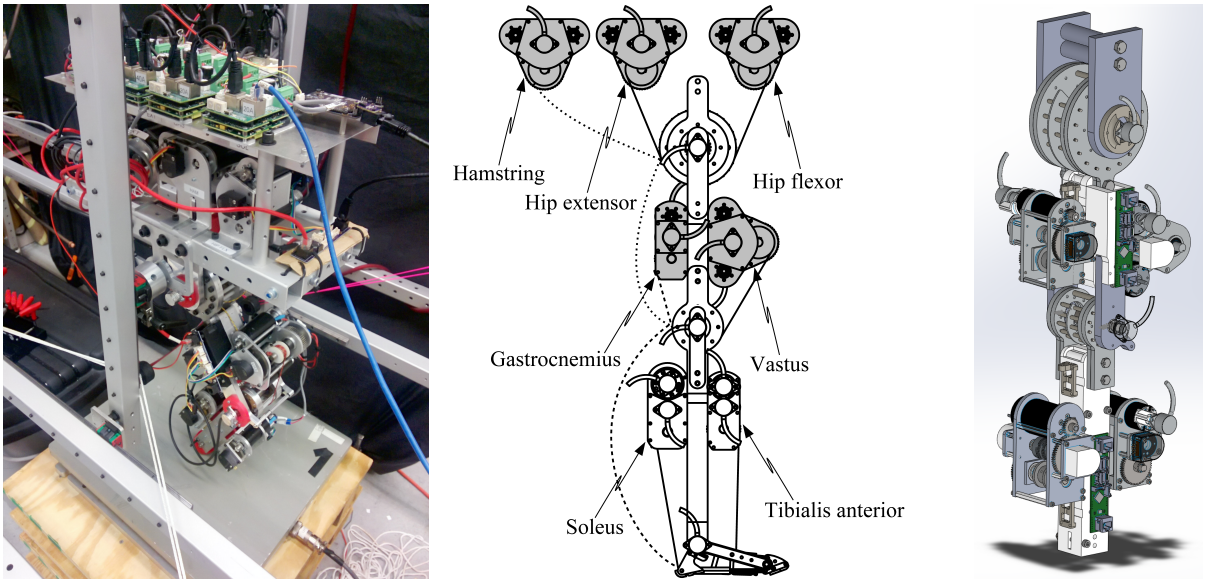
Chapter 3

RNL3 Hardware Development

As an ongoing effort of evaluating neuromuscular controllers, a series of standalone hardware platforms named Robotic Neuromuscular Leg (RNL) have been developed. All RNLs are 1:2 *dynamically scaled* [21] planar legs (constrained on their sagittal planes), with physical dimensions and designed inertial properties based on research in 2D humanoid walking dynamics [6]. RNL1 was designed to verify the feasibility of achieving actuation capabilities similar to human legs in a dynamically scaled leg; it consists of a fixed thigh and a moving shank connected by a knee joint, and 2 antagonistic Series Elastic Actuators (SEAs) driving the joint through a cable-drum mechanism. With the capabilities verified, RNL2 was developed upon the basis of RNL1, with the thigh element connected to a fixed cage (as trunk element) through a hip joint actuated in the same way as the knee. It enabled verification of various swing-leg controllers [20] including decentralized ones such as [4]. However, the joint design of RNL2 was inherently incapable of supporting bi-articular SEAs needed to verify neuromuscular swing-leg control. Also, due to the fixed trunk element and therefore fixed hip location, it cannot support stance-leg behavior. RNL3 was designed to address both issues with a free trunk element and a bi-articular hamstring

SEA. This chapter discusses the details of its design.

3.1 Mechanical Design



(a) full assembly (b) actuation schematic (c) leg 3D model

Figure 3.1: Overview of RNL3 mechanical design

The main mechanical structure of RNL3 consists of a segmented leg, a trunk segment, and a fixed cage. The trunk segment is mounted on the cage through passive joints allowing full 3 DoF in the sagittal plane. 2 sets of sliders implement the translational DoF and a pair of bearings implement the rotational DoF. Each DoF can be individually removed by locking the corresponding joint. The trunk houses power and sensing electronics (section 3.2) and 3 hip actuators (section **??**). The leg consists of thigh and shank segments joined by a knee joint, and is mounted on the trunk segment through a hip joint. The following sections discuss the details of this mechanical design.

3.1.1 Segmented Leg

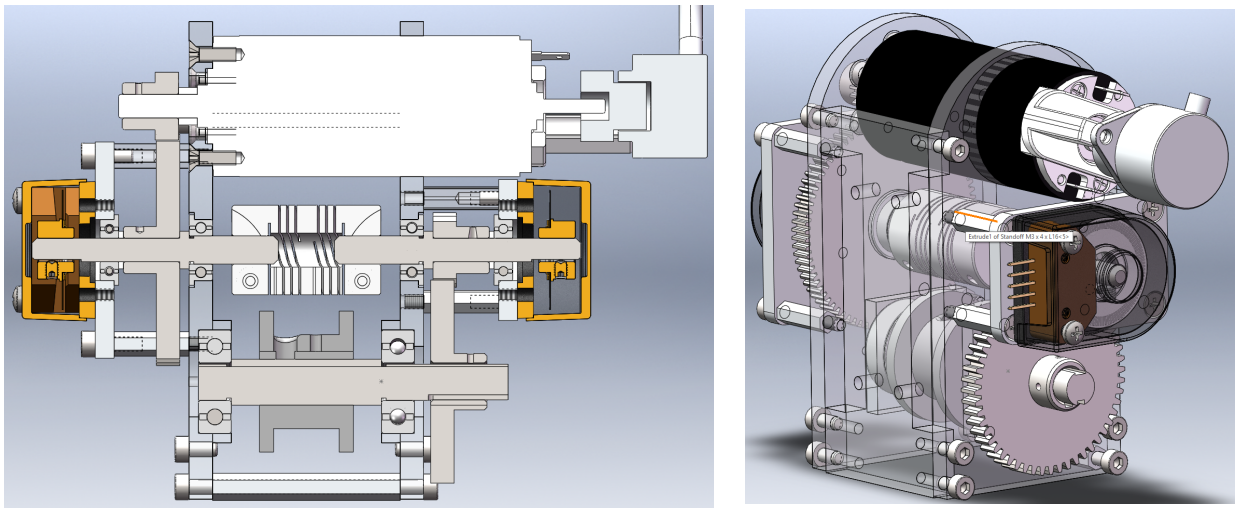
Fig. 3.1b illustrates the general structure of the segmented leg and its actuation scheme. In this 3-segment configuration (thigh, shank, and foot), 7 series-elastic actuators (see Section 3.1.2) drive 3 joints (hip, knee, and ankle) antagonistically through vectran fiber cables. Each SEA extends or flexes joints in the same way as their muscle group counterparts, as shown in Table 2.1. A SEA cable originates from the output drum of the SEA, passes through rollers on the joint it spans (two joints in the case of bi-articular SEA, see section 3.1.3), then ties onto its anchor point on segment. Hip actuators (GLU, HFL, HAM) are mounted on the trunk segment (not illustrated), knee actuators (VAS, GAS) on the thigh, and ankle actuators (SOL, TA) on the shank. This actuation scheme closely resembles how muscle-tendon units are positioned, routed, and connected in human legs. It also results in a weight distribution similar to human legs.

Fig. 3.1c shows the 3D mechanical assembly model of the leg. The foot segment and the ankle joint are currently not used and therefore left unimplemented; they can be assembled onto the end of shank in the same way as other joints in a future iteration. In stance experiments, a point foot made of polyurethane is bolted to the end of the shank to provide shock absorption. Despite lack of ankle actuation, the ankle SEAs remain assembled in order to maintain weight distribution. 40x40 square-profile aluminum extrusion tubes serve as the main structural element of both thigh and shank segments. SEAs (Section 3.1.2) and sensing electronics (Section 3.2.1) are mounted on segments through screws.

3.1.2 Series-Elastic Actuators

All 3 generations of RNL employ electrically-powered series-elastic actuators. The 7 SEAs used in RNL3 are custom self-contained modules designed around DC brushed motors (RE

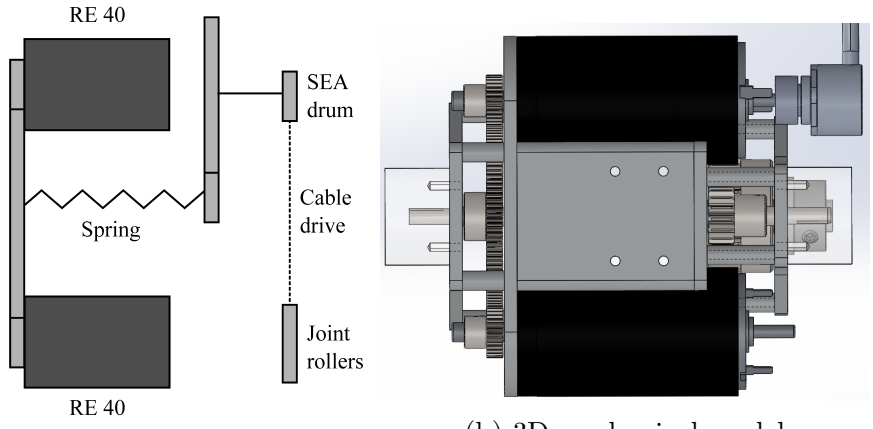
series by Maxon Motor ag). They are capable of exerting and closely tracking given desired cable tension profiles. They share a common drive-train design but differ in size: GLU, HFL, HAM, and VAS are the most powerful variant, each carrying two RE40 motors in parallel (both mechanically and electrically). SOL carries one RE40 motor, while GAS and TA each carries one RE30 motor. These motors are chosen to satisfy both torque and speed design targets while fitting within weight budget (see 3.1 for whole robot weight goals). Specifically, the dual-motor configuration allows a more compact actuator design than that using a single larger motor providing the same total power.



(a) (b)

Figure 3.2: 3D model of GAS SEA design

Fig. 3.2a is a section view of the GAS SEA, and therefore the common drive-train design shared by all SEAs. It incorporates two gear reduction stages coupled by a linear torsion spring. The first stage ("motor stage") couples the motor shaft with one end of the spring with a gear ratio of 4:1. The second stage ("load stage") couples the other end of the spring with the cable drum with a gear ratio of 3:1. Considering the relatively low load and reduction ratio target, spur gears are used to implement both stages. The torsion spring is implemented with a flexible cut-beam shaft coupler, which is rated well above the maximum torque produceable by the motor, has a suitable stiffness value range, and accomodates



(a) schematic

(b) 3D mechanical model

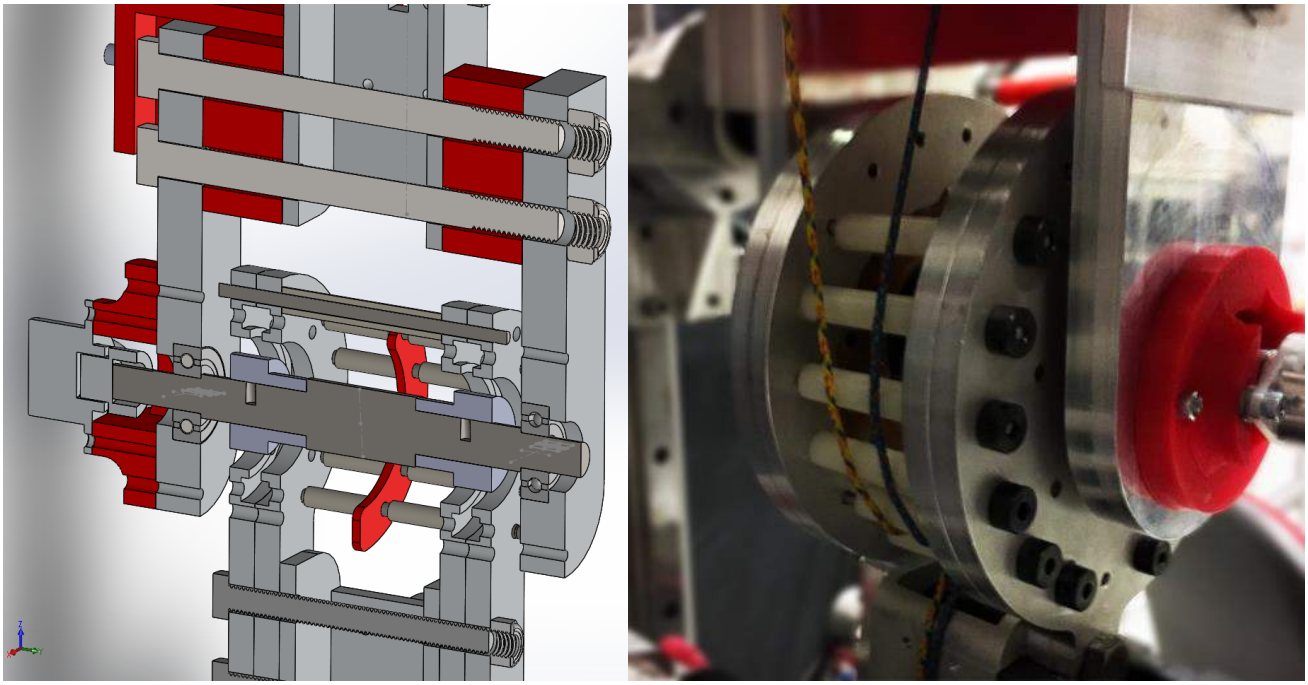
Figure 3.3: Dual-motor SEA drivetrain

mis-alignment between shafts within manufacturing and assembly tolerances. The torque exerted by the spring on the output stage can be indirectly measured by measuring its deflection, which can in turn be measured using two rotary encoders attached at both ends of the spring. A third encoder is attached directly to the motor shaft to provide velocity feedback for the motor controller (section 3.2).

Fig. 3.2b shows the overall mechanical structure of the GAS SEA. The housing consists of 5 machined aluminum alloy plates (rendered translucent in figure) joined with screws and spacers. Each gear stage consists of 2 shafts (including motor shaft) mounted on precision ball bearings, with spur gears positioned at shaft steps and secured with set screws.

The SOL design is simply an up-sized version of that of GAS, with larger physical dimensions to accomodate for the larger motor, spring, and gears. Hip actuators and VAS, however, feature a slightly more complex design as they incorporate two motors in parallel. Fig. 3.3 illustrates their drive-train. The two motors are mechanically coupled through the first gear stage, providing 2x torque compared to SOL.

3.1.3 Joint Design



(a) knee joint 3D model section view (b) hip joint photo

Figure 3.4: RNL3 joint design

While it is common for a robotic segmented leg to have rigid revolute joints, joints in human and animal legs are compliant thanks to soft tissue. The RNL series was designed to capture such compliant joint dynamics by introducing a floating joint design, allowing limited translational compliance through molded rubber. Also, the joints need to accomodate for cable actuation. Fig. 3.4 shows the mechanical design of RNL3 joints. The knee joint (Fig. 3.4a), as an example, joins the thigh and shank. A pair of aluminum plates are bolted onto the thigh segment, each housing a roller bearing. A shaft is supported on the pair of bearings, forming an ordinary revolute joint. The compliance is implemented on the shank side of the joint: Similar to the thigh side, two aluminum plates are bolted onto the shank segment. Molded rubber (Fig. 3.5) fills the cavity between the plates and a spur gear located at the center. The rubber element essentially forms a planar spring, giving the gear limited translational DoF within the plate enclosing it. The whole

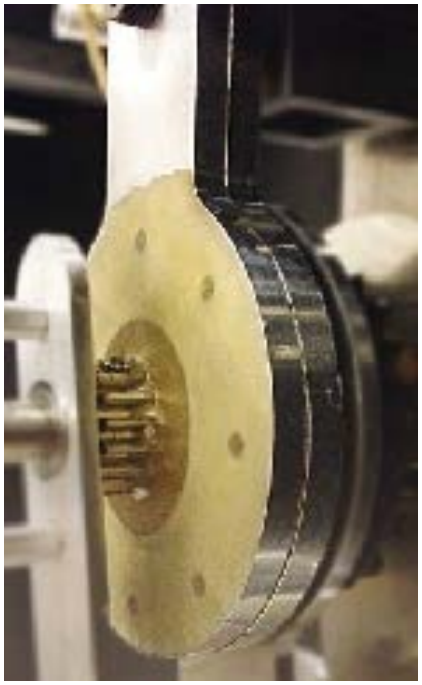


Figure 3.5: Molded rubber element and spur gear within joint plate

compliant joint is completed by rigidly attaching the gears onto the shaft, which combines the rotational DoF with the spring-loaded translational DoF. This particular layout not only achieves a compact and simple design; it also simplifies mounting of an absolute encoder for measuring joint angle: the encoder head attaches to the end of the shaft, while the encoder body screws onto the plates housing the bearings.

The joints provide a constant moment arm for both mono- and bi-articular SEA cables using a number of rollers in a squirrel-cage configuration, as shown in Fig. 3.4: Each roller consists of a steel rod and a nylon sleeve over it. Rollers are attached onto the distal part of the joint through miniature shaft collars. A cable coming from SEA wraps around the exterior of the roller cage as shown in Fig. 3.6. When the cable is under tension, a unit length linear motion of the cable translates to a unit length arc motion of the roller cage. This holds true independently for both joints in a bi-articular configuration: suppose the knee joint is fixed, then the distal section of the cable past the last roller on the hip

becomes essentially a part of the thigh-knee-shank rigid body, which means the mechanism degenerates into a mono-articular cable attached through hip to thigh only; on the other hand, suppose the hip joint is fixed, since the hip rollers do not impede pulling motion on the cable, the system degenerates into a mono-articular cable attached through knee to shank only. When neither joints are fixed, cable tension is distributed between actuating the two joints, but the respective moment arms remain the same.

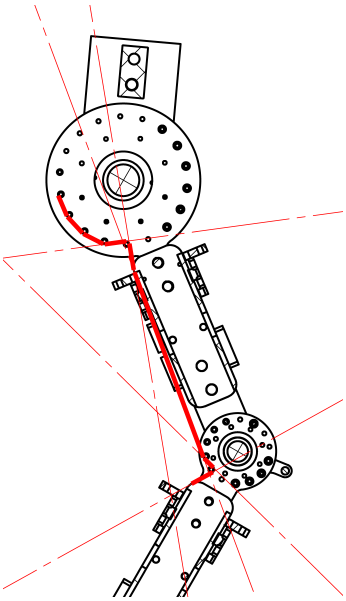


Figure 3.6: Bi-articular cable actuation

3.2 Electronics Design

The RNL3 consists of several software and hardware subsystems as shown in Fig. 3.7. A PC is dedicated as the real-time controller of the system, running control software on the Simulink Real-time (SLRT, by Mathworks Inc) platform (Fig. 3.8). Another PC ("host PC") running MATLAB/Simulink (Mathworks Inc) on full desktop OS creates and manages the software on the controller PC. In order for the controller to access actuators and sensors in the robot, interfacing electronics is needed. All actuators in the system are DC

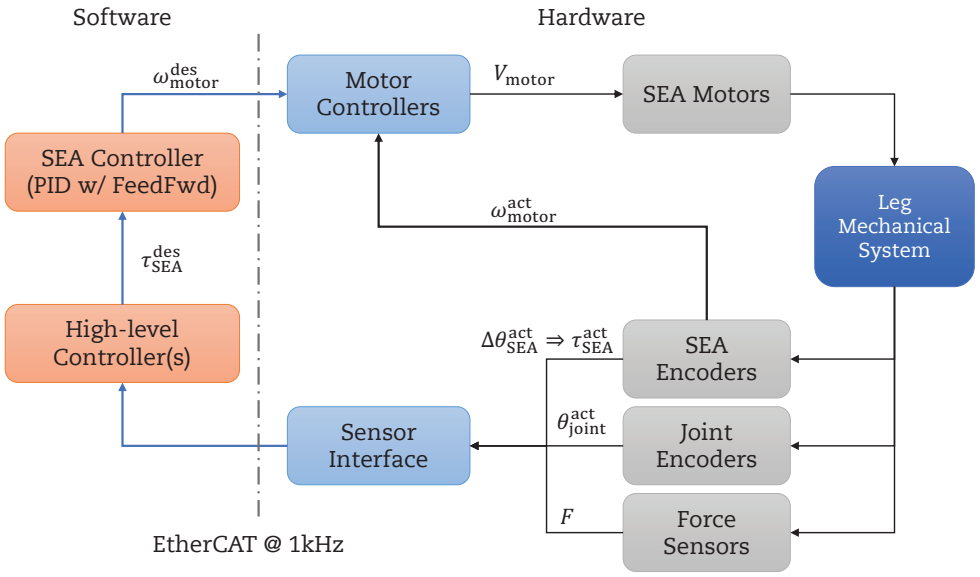


Figure 3.7: RNL3 top-level system block diagram

motors, which can be managed by COTS (commercial off-the-shelf) motor controller units (DZEANTU series by AMC). Sensors in RNL3, on the other hand, are relatively diverse and are spread out across the robot. Each SEA are equipped with 3 incremental encoders (section 3.1.2): 2 for measuring spring deflection and therefore torque/force output (EM2 by US Digital), 1 for motor feedback (RM22I by Renishaw). The motor controller directly accepts the motor encoder, but the spring encoders need to be separately interfaced. There is also an absolute encoder (RM22S by Renishaw) mounted on each joint measuring its angle. SEA cable tension sensors was envisioned and partially designed [1]; they utilize force-sensitive resistors (FSRs, FlexiForce by TekScan) and require analog signal conditioning. Recently, as a part of preparing RNL3 for stance experiments, a standalone 6DoF ground reaction force (GRF) instrumentation platform (aka. "force plate", OR6-7-2000 by AMTI) has been added to the system. In the full 3-segment configuration, there are in total 14 incremental encoders, 3 absolute encoders, and 7 FSRs. Even in current 2-segment configuration, there are 10 incremental encoders, 2 absolute encoders, and potentially up

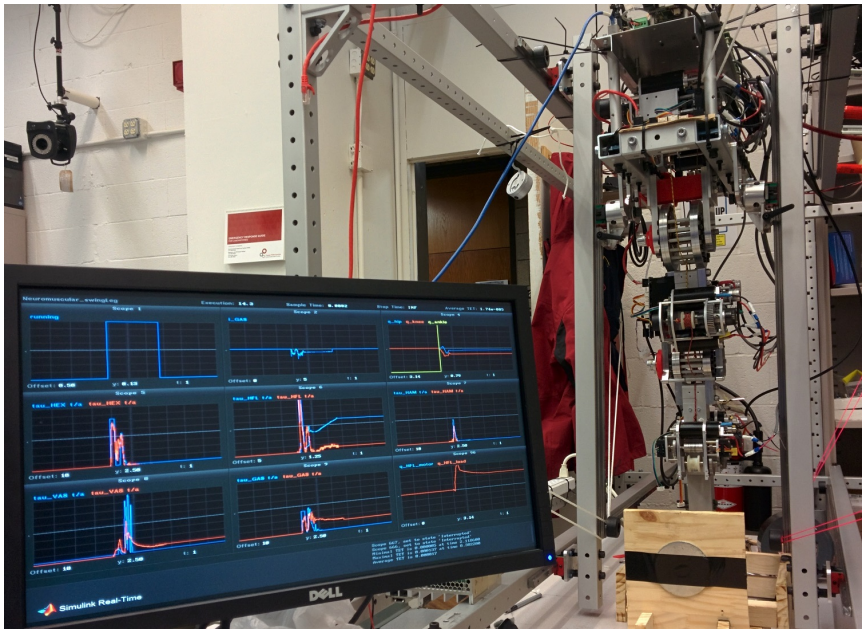


Figure 3.8: Simulink Real-time and RNL3 (shown running neuromuscular swing controller experiments)

to 5 FSRs.

All motor controller units communicate with the controller PC via a EtherCAT bus, a high-speed real-time fieldbus which guarantees constant 1 kHz full-duplex update rate. On the other hand, there was no COTS solution available that can simultaneously interface with all sensors. Also, routing of sensors, especially those located in the distal parts of the robot, presents a challenge due to long cables susceptible to mechanical interference, as well as electrical noise from nearby motor power cables. Instead of directly routing the sensors to central processing, RNL3 employs a de-centralized sensor interfacing scheme which matches the overall design of the robot: sensors are connected through minimal cabling to a nearby "sensor hub" PCB (section 3.2.1), which relays collected data through a local daisy-chaining bus to a central EtherCAT slave node module (section 3.2.2). The force plate provides an RS-232 serial interface covered in section 3.2.3. The whole system is connected as shown in Fig. 3.9.

[1]They are not currently used in control, therefore not mounted in hardware.

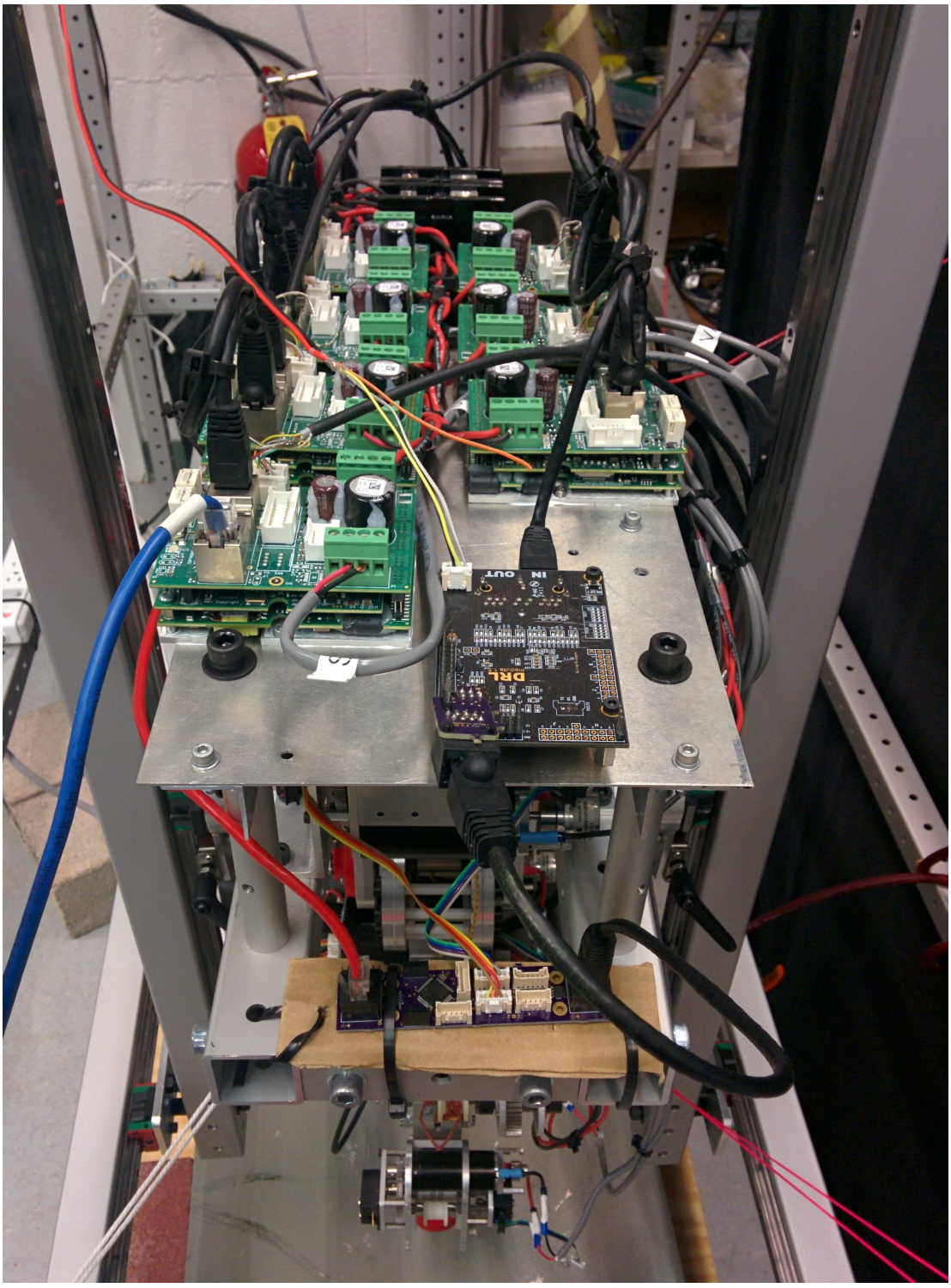


Figure 3.9: RNL3 actuator-sensor interface connection

3.2.1 Sensor Hub

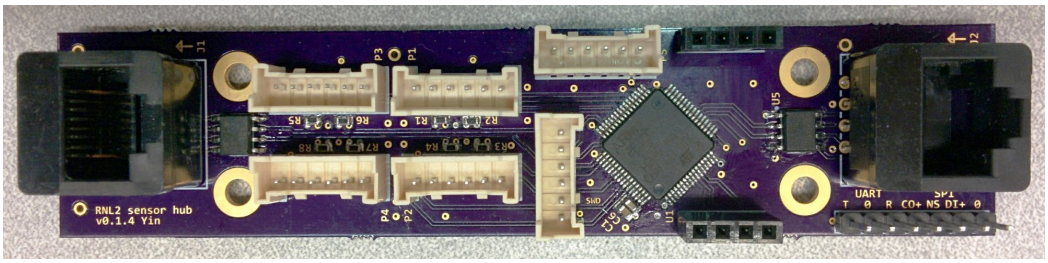


Figure 3.10: Sensor hub PCB assembly (early iteration)

The sensor hub is a custom PCB with an embedded processor (STM32F100R8T6 by ST) that integrates power supply and signal conditioning for 4 incremental encoders, 1 absolute encoder, and up to 4 analog signals. It provides a custom daisy-chainable differential-signal SPI (Serial Peripheral Interface) bus interface, which allows reading data from multiple sensor hubs using only one port.

The PCB is designed to accept either either single-ended or differential signaling incremental encoders, through optional differential receivers. The standard single-ended quadrature encoder signals are connected to built-in decoder hardware blocks on the processor, which keep one 16-bit counter value for each encoder. The RM22S encoder provides a standard differential SSI (synchronous serial interface) slave port, which connects to a compatible SPI master port on the processor through differential transceivers. The sensor hub PCB itself does not host any analog signal processing; instead, it accepts 4 analog inputs in the 0∼3.3 V range, which are connected to the built-in 12-bit ADC (analog-to-digital converter) on the processor. FSRs can be interfaced through a small daughter PCB containing analog signal conditioning circuits.

The physical bus consists of differential clock and data lines, a data request line, and both +5 V and +3.3 V power supply lines. Each module provides an upstream port (left side

in Fig. 3.10) and a downstream port (right side), both of which using RJ45 jacks. When a sensor hub detects a data request from the upstream port, it first sends out readings of all sensor channels upon clocks at the upstream port, then starts relaying data from its downstream port. In this way, reading data from the "upmost" upstream port in a daisy-chain of sensor hubs results in data from all sensor hubs, in the order of increasing distance from the reading port. In other words, the chain is conceptually equivalent to a series of shift registers with shared clock, daisy-chained data in/out lines, and shared asynchronous parallel-load line which corresponds to the data request line. Implementation of such behavior is as follows: The processor sets up a full-duplex SPI slave port which responds to data request and clock signals from the upstream port, both of which also relayed to the downstream port. At each clock cycle, the processor sends a bit to the upstream port and receives a bit from the downstream port. Incoming data from downstream is first buffered when the processor sends out its own sensor readings; it is then relayed upstream.

3.2.2 EtherCAT Slave Node

Figure 3.11: Medulla EtherCAT slave module

While the sensor hub exposes all sensor data on a single SPI port, it cannot be read by the controller PC directly, as the PC does not have an SPI port. Since there is already

an EtherCAT bus between the controller PC and motor controllers, sensor data can be "bridged" onto the EtherCAT bus. Developing a functional EtherCAT slave node is non-trivial, but fortunately a general-purpose EtherCAT slave node module named Medulla was already developed for and verified on the Atrias robot [18], and it contains an embedded processor (ATXMega128 by Atmel) which can be reprogrammed as the bridge module: A firmware was written to read from a SPI master port on the processor and transfer the data onto the EtherCAT bus as periodically requested by the bus. Sensor hub chain connects to this SPI port through differental transceivers.

3.2.3 Force Plate Interface

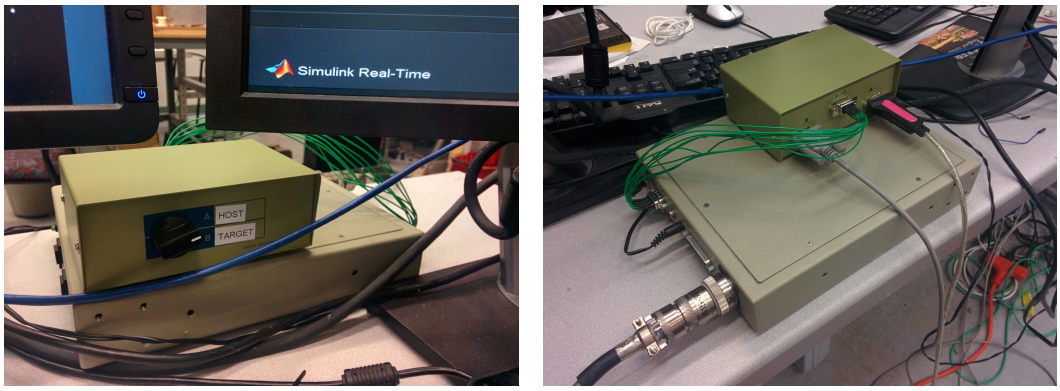


Figure 3.12: Force plate amplifier and switch connection

The force plate is directly interfaced with a matching calibrated amplifier unit (MSR-6 by AMTI). This amplifier provides both amplified analog outputs, and an asynchronous[2] RS-232 serial port for configuration and digitized data. Since the force plate is used only for instrumentation and experiment sequencing purposes, no real-time guarantee is needed, so it is acceptable to directly connect the controller PC to the amplifier through the serial port.

The serial port on the amplifier accepts a simple binary command-response protocol. Com-

mands include starting and stopping data streaming, changing the data rate, and software zero offset calibration. While it is desirable to sequence the commands in Simulink Real-time on the controller PC, the process involves a change in serial port data rate, and therefore cannot be handled by Simulink Real-time. As a workaround, an RS-232 multiplexer is added to switch the amplifier between connected to either the controller PC or the host PC (Fig. 3.12). Since the host PC runs full desktop OS, it is possible to write programs in a imperative language that handles the initialization sequence. Once initialized, the amplifier continuously transmits data with a simple format at a known fixed rate, which can be handled on the controller PC using relatively simple Simulink code.

3.3 Tuning

3.3.1 Motor Controller Unit

The motor controllers were tethered to a PC to have its internal control loop parameters tuned. They were tuned with the motors connected to them and assembled on un-loaded SEAs. The motor current tracking loop was automatically determined according to motor datasheet. The velocity loop was tuned by using a 1 Hz square wave signal as velocity command and observe the step response plot of motor as gains are tuned. Tuning was stopped once the step response plot shows reasonable response time and overshoot.

[2]Both line-level (RS-232 is not clocked) and frame-level (sampling not synchronized with controller)

3.3.2 SEA Controller

SEA controllers (section 4.1.2) were ported to Simulink Real-time and run with hardware in the loop. The feedforward gain k_{ff} is set to 0.75 (same as in simulation), and then the PID gains are tuned on hardware: The trunk is locked in place and the SEAs are fully assembled onto the robot. For each SEA, a 1 Hz sinusoid was applied as desired torque profile. The torque tracking response was monitored as the gains were adjusted. Some torque oscillation was tolerated to achieve faster response time and reduce torque tracking error.

3.4 Preliminary Hardware Experiments

3.4.1 Working Around Hardware Limitations

A flaw in hardware design was not discovered until the vertical movement of the trunk segment was unlocked. Two orthogonal groups of slider rails provided the translational DoF for the trunk. Although this appears valid kinematically, within each group the sliders were not properly synchronized; as a result, these overconstrained prismatic joints jammed easily and produced inconsistent friction forces of significant magnitude, both static and dynamic. A temporary solution was conceived to allow a limited range of approximately vertical motion of the trunk using a boom-like structure as shown in Fig. 3.13. An aluminum tube in the sagittal plane has one end attached to the cage through a revolute joint, the other end rigidly attached to the trunk. While the trunk is now on an arc, the radius of this arc is large enough for the limited range of motion to be approximately vertical. Friction at the revolute joint is negligible. Also, in order to prevent the robot

from fully collapsing onto itself, a hard limit on boom position was created using strong ropes. When the boom is too low, the rope becomes tout and starts to support the weight of the whole robot.

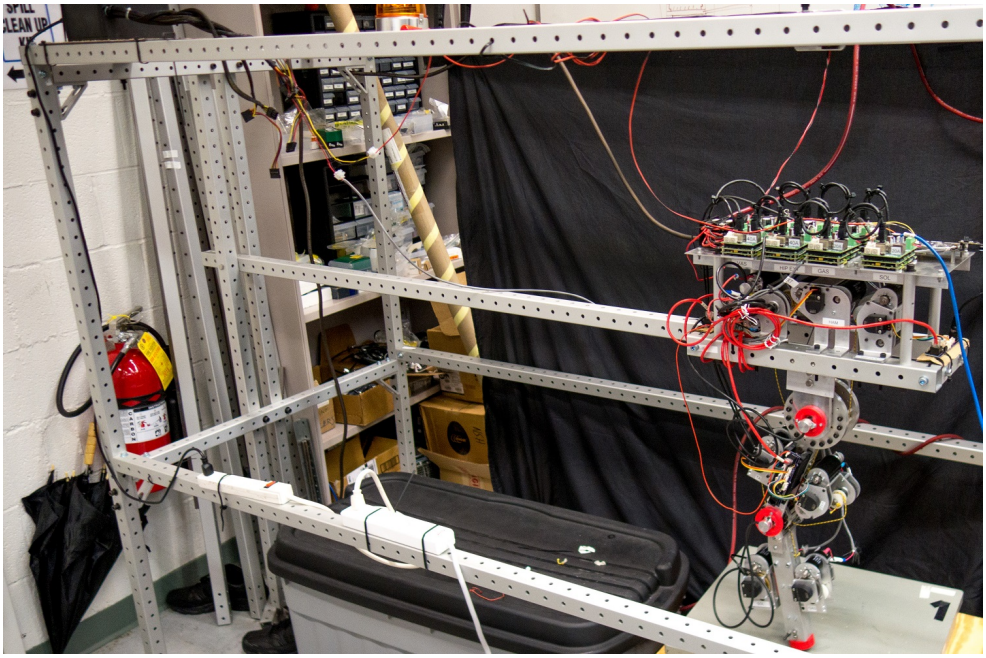
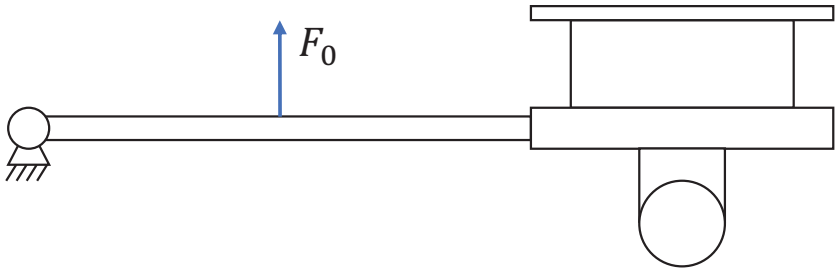


Figure 3.13: Photo of modified RNL3 mounting setup

Another problem became apparent soon: the leg far exceeded its design weight requirements. Table 3.1 compares the actual weight to dynamically scaled human value. This proved challenging for the SEAs to even just balance the gravity, let alone realizing bouncing gait, as they were designed with original weight target in mind. Reduction of mass was not possible at the time because it would have required a complete mechanical redesign. As an alternative, a constant force spring was attached to the boom as shown in Fig. 3.14. This effectively provides an adjustable reduction in total gravity on the robot, and reduced the load on the SEAs.

(a) Photo



(b) Equivalent effect on mechanism

Figure 3.14: Constant force spring

Table 3.1: RNL3 weight comparison [kg]

Segment	Human	Scaled/Target	Actual
HAT (trunk)	53.50	13.38	8.30
Thigh	8.50	2.13	3.57
Shank	3.50	0.88	2.28
Total	65.50	16.38	14.15

3.4.2 Trial Runs

In the initial trial runs, the robot was powered on in a maximally collapsed configuration (as allowed by the hard limit described in section 3.4.1), with the force plate disabled (switched to host PC) in the beginning so that no high-level controller was active (section 4.1.3) and only low-level SEA controllers were active and maintaining cable tension. This tension was verified by plucking the cables slightly sideways. Then the force plate was enabled (switched to real-time controller PC) and the neuromuscular stance controller became active. The robot was able to stand up. However, the leg was slowly overextended

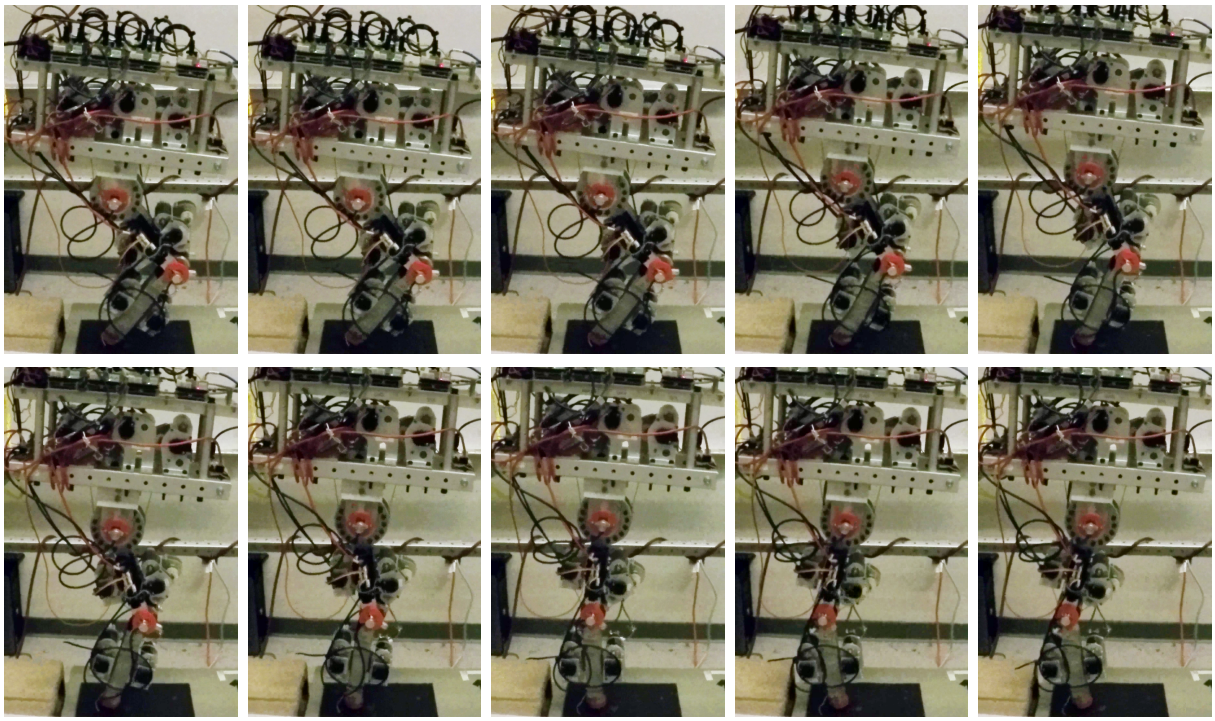


Figure 3.15: Frames from video recording of one hardware trial run; real time difference between adjacent frames: $\Delta t = 1/10\,\mathrm{s}$

despite VAS powered off and GAS actuating, as shown in Fig. 3.15. Data captured from the SEA controller (Figs. 3.16, 3.17) indicated that VAS SEA motors were saturated, while the desired GAS SEA torque from stance controller apparently was far insufficient to prevent over-extension.

Unfortunately, the robot proved to be not durable enough for repeated experiments and a variety of mechanical failures occured frequently. Despite efforts to repair and strengthen the hardware, eventually hardware experiments could no longer be conducted. Analysis of failure modes, design issues, and possible solutions are discussed in section 5.2.

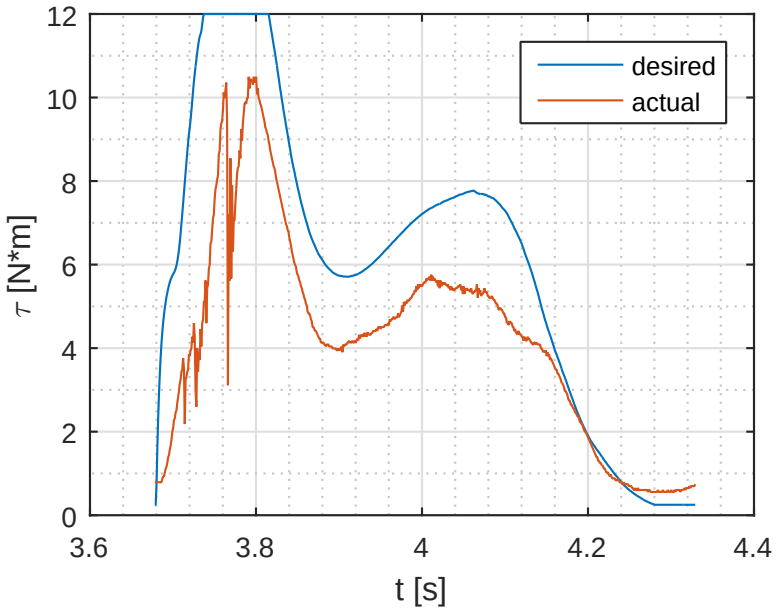


Figure 3.16: VAS SEA torque tracking

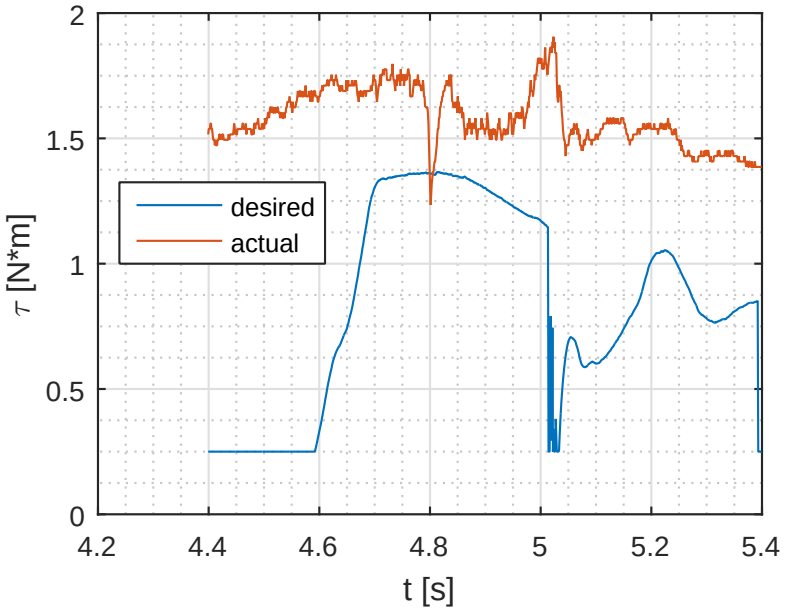


Figure 3.17: GAS SEA torque tracking

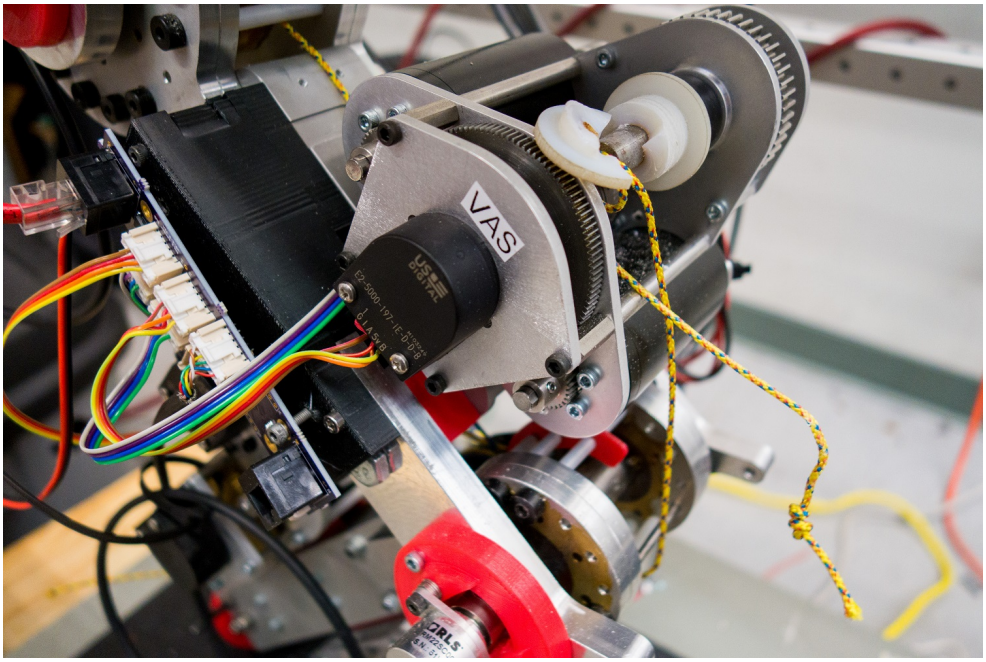


Figure 3.18: VAS SEA output drum and cable failure

Chapter 4

Simulation

In order to validate the design of the controller and determine initial values for its parameters, a simulation model was developed in Simulink environment. It consists of a physical model of the robot and its environment, and a complete implementation of controllers. Design and implementation of these systems are detailed in section 4.1.

4.1 Model Development

Fig. 4.1 shows the top-level organization of the simulation model, which follows the subsystem boundaries of the actual robotic system shown in Fig. 3.7. The following sections discuss the implementation of each subsystem.

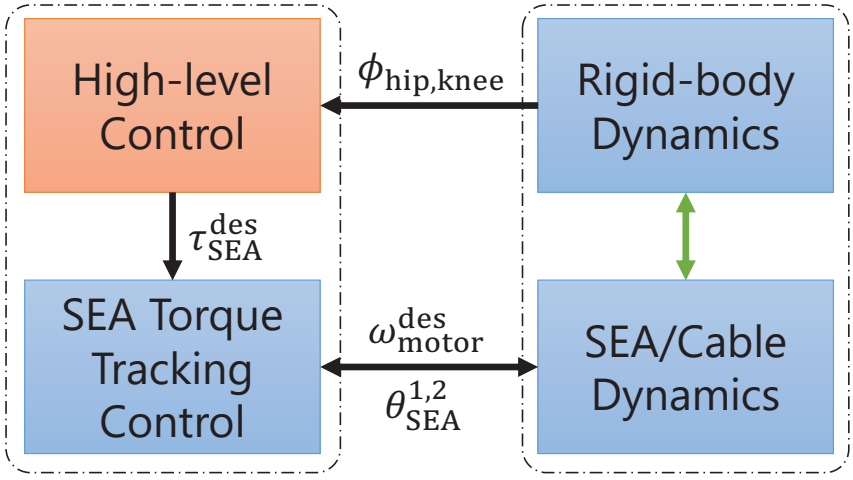


Figure 4.1: Simulation top-level schematic

4.1.1 Physical Model

For convenience of modeling, the physical system of the robot and its environment can be decomposed into the following subsystems:

1. Rigid body open kinematic chain consisting of cage, trunk, leg segments, and all joints in between
2. SEA mechanics
3. Actuation cable mechanics
4. DC motors and their controller units
5. Ground contact of point foot (section 3.1.1)

The rigid body dynamics is modeled using the SimMechanics blockset in Simulink. Geometry (for visualization) and inertial properties are imported from SolidWorks into SimMechanics body blocks. Fig. 4.2 shows the assembled SimMechanics model. Some details of graphics are omitted to avoid visualization problems; such omission is purely visual and does not change the inertial properties of the underlying rigid bodies. Two additional

grounded bodies are added as visual references: a ground plane, and a thin stick indicating the vertical constraint on trunk.

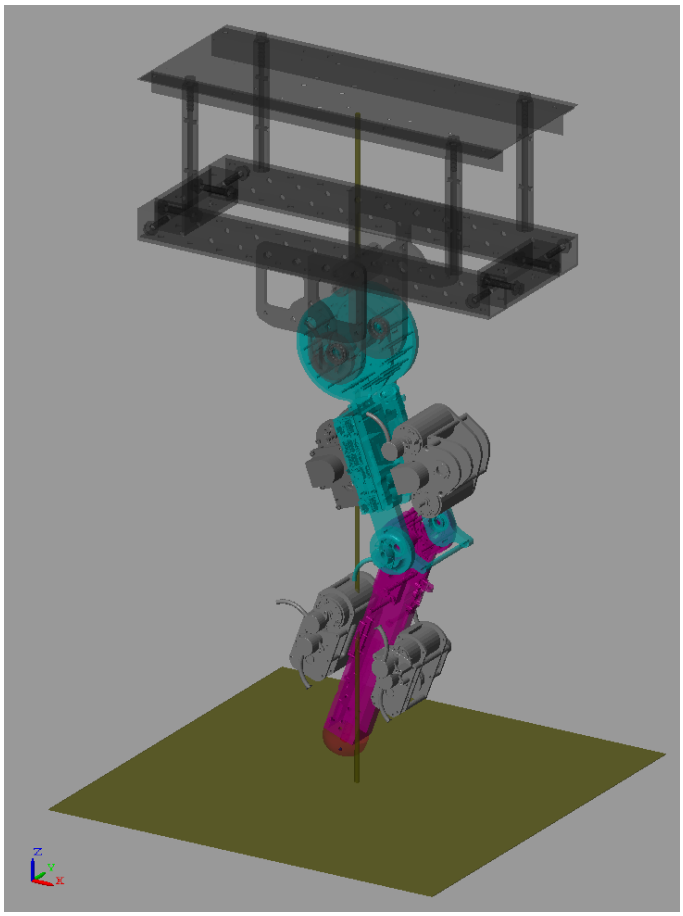


Figure 4.2: Simulation top-level schematic

While SEA mechanics were initially modeled using SimMechanics and simple linear torsion spring model ($\tau = k\Delta\theta$), isolated testing indicated that reaction force (excluding gravity) and moment caused by actuation between an SEA and the leg segment it is mounted to is negligible (force near rounding error, moment $< 0.05\,\mathrm{N\,m}$). This means it is possible to speed up the simulation speed by replacing the 3D rigid body dynamics with 1D equations of motion. The SEA drivetrain can be seen as 2 rotating bodies coupled by a torsion spring: one being the lumped sum of all bodies rigidly connected to the motor (see below for modeling of dual parallel motors), the other to the output drum. Let subscript m

denote the motor, s the spring (sf for its motor-side and sr for its load-side), and p the output drum ("pulley"). Equations of motion are as follows:

$$J_1\ddot{\theta}_m = \tau_m - \tau_s/i_1 \tag{4.1}$$

$$J_2\ddot{\theta}_p = \tau_s \cdot i_2 - \tau_p \tag{4.2}$$

$$\tau_s = k_s(\theta_m/i_1 - \theta_p \cdot i_2) \tag{4.3}$$

$$J_1 = J_m + J_{\mathrm{s1}}/i_1^2 \tag{4.4}$$

$$J_2 = J_{\mathrm{s2}} \cdot i_2^2 + J_p \tag{4.5}$$

These equations can then be integrated using only Simulink primitives.

Since mass of vectran actuation cables are negligible, they are also modeled as lumped parameter models. According to the datasheet, these cables exhibit linear strain-stress relationship when taut. Therefore they can be modeled as single-sided linear springs. An additional empirical damping term is added to model inelastic losses and help maintain numerical stability of simulation. Length, or rather slack of each cable is calculated using position of joint(s) and SEA output drum:

$$\Delta l = r_j \cdot \Delta\theta_j - r_p \cdot \Delta\theta_p \tag{4.6}$$

Cable tension is calculated as follows:

$$F_c = \begin{cases} \max\left(k_c\Delta l + d_c\dot{\Delta l}, 0\right) & \Delta l < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

The force is converted to torque applied at both the joint(s) and SEA output drum:

$$\tau_j = r_j \cdot F_c \tag{4.8}$$

$$\tau_p = r_p \cdot F_c \tag{4.9}$$

The motors powering the SEAs are all brushed DC motors. Assuming no magnetic loss, their electro-mechanical response follow the following equations:

$$V - iR - L\dot{i} - k_B\dot{\theta_m} = 0 \tag{4.10}$$

$$\tau_m = k_B i \tag{4.11}$$

where V, i, R, L are voltage, current, resistance, and inductance of armature respectively; k_B is the back-EMF constant of the motor, which is equal to the torque constant of the motor (no magnetic loss); θ_m is rotor angle; τ_m is the output torque of motor. In the case of two identical motors connected mechanically and electrically in parallel, since they share the same voltage and rotor angle, according to above equations, current passing through them and their torque output are also the same. Therefore total torque and current are both double of those of a single motor.

The AMC DZE-series motor controllers used to drive these motors are set in velocity closed-loop control mode. According to datasheet, the velocity control loop runs at 5 kHz, while desired motor velocity input can be updated at 1 kHz through EtherCAT bus. As a simplification, the whole motor controller can be modeled as a discrete-time PID controller tracking desired motor velocity with motor voltage as output. Using built-in PID loop tuning tool in Simulink, the controller parameters are chosen to achieve response time and overshoot similar to those observed when tuning the motor controller hardware with motor(s) assembled in an SEA without load.

Interaction between the point foot mounted on shank and the ground follows the same model described in Appendix IV of [6]: the vertical component of the GRF is modeled as

a nonlinear spring-damper that is capable of capturing partially elastic impact behavior; the horizontal component incorporates a simple state machine that generates either static or coulomb friction behavior. Both components are only active when the foot point has positive penetration into ground ($z < 0$).

4.1.2 Low-level SEA Torque Controller

SEA controllers regulate motors to track given desired torque profiles at the output drum of each SEA. The controller implemented is inspired by described in [30]. It takes desired SEA torque at pulley τ_p^{des} and outputs desired motor angular velocity ω_m^{des}. Both feedback and feed-forward terms are used to ensure stability and tracking accuracy under variable loading conditions. The feedback term is a discrete-time PID loop on torque tracking error $\tau_p^{\text{act}} - \tau_p^{\text{des}}$. The feed-forward[1] term is based on simplified SEA dynamics described in 4.1.1. By taking derivative of the linear spring equation and substituting variables:

$$\dot{\tau}_p = i_2 k_s (\omega_m / i_1 - \omega_p \cdot i_2) \tag{4.12}$$

Assuming that the feed-forward term is solely responsible for generating output torque, then in order to track a continuous torque profile, motor velocity should be:

$$\omega_m = \frac{i_1}{i_2 k_s} \cdot \dot{\tau}_p + i_1 i_2 \omega_p \tag{4.13}$$

While ω_p is not directly known, it can be estimated using the load-side spring shaft encoder reading θ_{sr}:

$$\omega_p = \dot{\theta}_{\text{sr}} / i_2 \tag{4.14}$$

Therefore the full feed-forward term:

$$\omega_m^{\text{ff}} = \frac{i_1}{i_2 k_s} \cdot \dot{\tau}_p + i_1 \dot{\theta}_{\text{sr}} \tag{4.15}$$

Taking into consideration of sensor data propagation delay ($> 1\,\text{ms}$) and noise from numerical derivative, the feedback term is attenuated by a constant factor before adding to the final desired motor velocity command:

$$\omega_m = \omega_m^{\text{PID}} + k_{\text{ff}} \omega_m^{\text{ff}} \tag{4.16}$$

Also, in order to prevent the actuation cables from loosening, a lower limit is applied to each desired torque profile signal. This ensures that all cables remain taut when the robot is running.

4.1.3 High-level Controllers

This subsystem takes feedback signals from the robot model and takes Separate high-level controllers are active for stance and flight phases: neuromuscular bouncing controller during stance, and a conventional feedback controller during flight. In order to reliably determine whether current phase is stance or flight and therefore which controller is active, the vertical component of ground reaction force is used. A binary "whether in contact" signal is first derived from vertical GRF using a hysterisis switch. This signal then drives a state machine determining if a controller should be active:

1. Initially both controllers are off

[1]Strictly speaking it is also a feedback as it incorporates encoder reading, but since it does not compute error directly we shall still refer to it as "feed-forward"

2. On rising edge of contact signal (touchdown): stance turns on, swing turns off
3. On falling edge of contact signal (takeoff): if contact signal remains low (in flight) for a certain amount of time, stance turns off, swing turns on

The hysterisis and the delay mechanisms together act as a "debouncer" for the GRF signal, preventing spurious noise-induced controller switches.

Neuromuscular Stance Controller

As introduced in section 2.2, the neuromuscular stance controller consists of a virtual neuromuscular model and stance reflexes. This section discusses implementation details of the controller.

The stance reflexes are straightforward in implementation as they all consist of simple equations. Each Hill-type MTU is implemented as integration of the ODE (2.9) describing its dynamics. MTU length and moment arm calculations are also directly implemented as described in section 2.2.2. Mapping from MTU force output to SEA desired torque profile is implemented as follows: Each MTU with an SEA counterpart of the same name is mapped to desired torque of that SEA (see Table 2.1 for muscle and SEA functions). The only MTU without corresponding SEA is the biarticular RF, whose effect must be realized by actuating HFL and VAS SEAs. Also, since the ankle is not actuated, the ankle muscles and SEAs are unused, and biarticular GAS degenerates to monoarticular BFsH, which means only one of them are needed. Since GAS has an SEA counterpart while BFsH does not, the name GAS is kept while it actually serves the role of BFsH. After mapping, desired torques are calculated in the following way:

1. Calculate torque produced by muscle at virtual skeleton joint: $\tau_M = F_M r_M$

2. Desired SEA output torque should produce the same amount of torque at corresponding RNL3 joint: $\tau_{\text{SEA}}^{\text{des}} = \tau_M \cdot r_{\text{SEA}}/r_{\text{joint}}$

Flight Controller

While the main focus of this work is on the neuromuscular stance controller, a controller for the flight[2] phase is necessary to verify the hypothesis that the whole system can maintain stable in-place bouncing. The goal of this controller is to regulate the leg after it takes off so that it reaches a suitable configuration before touchdown, as a preparation for the next stance phase. Although a neuromuscluar swing-leg controller for walking and running was already verified on the same hardware, it assumes a forward step and therefore does not apply to in-place bouncing. Instead, a PD controller with heuristics was developed. It contains two separate PD loops over respectively the angle-of-attack α and length l of the virtual leg, which is defined as the line segment between the center of the hip joint and the foot point (in the sagittal plane). A constant desired angle-of-attack is given as α^{des}, while the desired leg length α^{des} is variable and is determined by heuristics. The PD loops are as follows:

$$\tau_\alpha = k_\alpha \left(\alpha - \alpha^{\text{des}}\right) + d_\alpha \dot{\alpha} \tag{4.17}$$

$$\tau_l = k_l \left(l - l^{\text{des}}\right) + d_l (\dot{l} - \dot{l^{\text{des}}}) \tag{4.18}$$

Notice that the derivative of desired leg length is not 0. Heuristics for determining this length are based on the following objectives:

1. Desired leg length should correspond to a valid leg configuration; that is, feasible and free of mechanical interference.
2. During early ascent (right after take-off), the leg should not be over-extended, in

order to avoid re-contact with ground.

3. During late descent, the leg should hold its length to prepare for impact.

The leg length limitations are implemented as constant upper and lower bounds on the final l^{des} value. The other two objectives are defined on flight trajectory of the robot. While RNL3 is currently not equipped with inertial sensors, an estimate of its flight trajectory can be calculated in the following way: Under the simplification that the robot is a point mass at hip joint during flight, its trajectory is uniquely determined by its initial position y_0 and velocity $\dot{y}_0 = v_0$, both of which can be calculated through forward kinematics right before take-off when the foot is still in contact with ground. With the trajectory estimate, the apex time Δt_a can also be estimated as v_0/g. Ascent and descent sub-phases can then be defined as $t - t_0 < k\Delta t_a$ $t - t_0 \geq k_t \Delta t_a$, respectively, with constant $k_t > 1$ denoting a delay in switching from ascent to descent. The constraint during ascent becomes:

$$l^{\text{des}} \sin\alpha \leq y_0 + v_0(t - t_0) - \frac{g}{2}(t - t_0)^2 \quad , \quad t - t_0 < k_t \Delta t_a \tag{4.19}$$

The descent can be trivially implemented as:

$$l^{\text{des}} = l|_{t_0 + k_t \Delta t_a} \quad , \quad t - t_0 \geq k_t \Delta t_a \tag{4.20}$$

τ_α and τ_l calculated from above rules are abstract "torques" that need to be mapped to SEA desired torques. A trivial mapping would be:

$$\tau_{\text{hip}} = \tau_\alpha \tag{4.21}$$

$$\tau_{\text{knee}} = \tau_l \tag{4.22}$$

$$\tau_{\text{GLU}} = [\tau_{\text{hip}}]_+ / i_{\text{GLU}} \tag{4.23}$$

$$\tau_{\text{HFL}} = [-\tau_{\text{hip}}]_+ / i_{\text{HFL}} \tag{4.24}$$

$$\tau_{\text{VAS}} = [\tau_{\text{knee}}]_+ / i_{\text{VAS}} \tag{4.25}$$

$$\tau_{\text{GAS}} = [-\tau_{\text{knee}}]_+ / i_{\text{GAS}} \tag{4.26}$$

where $[x]_+ := \max(x, 0)$, and i_{SEA} is the speed ratio between SEA output drum and the (first) joint it actuates. Such mapping have a problem: motion of thigh and knee are coupled. Without resorting to nonlinear control strategies, observing that the knee joint angle at take-off is always close to full extension (180°) in the ideal 2-segment bouncing gait [8], a simple torque compensation is applied:

$$\tau_1 = \tau_{\text{hip}} + k_1 \tau_{\text{knee}} \tag{4.27}$$

$$\tau_2 = \tau_{\text{knee}} - k_2 \tau_{\text{hip}} \tag{4.28}$$

where constants $k_1, k_2 > 0$ are hand-tuned to achieve reasonable decoupling of the angle and length controls. Also, instead of using GLU for realizing hip extension, biarticular HAM is a natural extensor of the virtual leg as a whole [4]; changing GLU to HAM improved the controller.

4.2 Simulated Experiments and Results

With the complete simulation model of both the physical system and the controllers, a number of simulations are run to determine a suitable parameter set for each controller, then explore the behavior of the system as a whole. The following sections detail each distinct simulation conducted.

4.2.1 SEA Torque Controller Tuning

Each SEA controller has 4 parameters: feedforward gain k_{ff} and PID gains. The feedforward gain is empirically set at 0.75 to provide the majority of control input. Then, the PID gains are tuned through optimization: A separate simulation model isolating each SEA and its controller was created[3], with a variable frequency sinusoidal signal as desired torque profile. Cost function is evaluated as the root mean square of torque tracking error. The optimization routine used is Covariance Matrix Adaptation Evolution Strategy (CMA-ES, [13]).

4.2.2 Stance Reflexes Tuning

All tunable parameters in the neuromuscular stance controller are within the stance reflexes. They are as follows:

- Reflex gains $G^{\text{VAS}}_{\text{VAS}}$, $G^{\text{BFsH}}_{\text{BFsH}}$, $G^{\text{VAS}}_{\text{BFsH}}$, $G^{\text{VAS}}_{\text{knee}}$
- Input time delay Δt^{in}
- Pre-stimulations S^{VAS}_0, S^{BFsH}_0

Among these gains, value $G^{\text{VAS}}_{\text{VAS}}$ and Δt^{in} are critical to stance behavior. In order to find a suitable initial parameter set capable of completing the stance phase, the full controller-robot simulation system is optimized upon. When the simulation starts, the robot falls freely from an initial configuration that resembles apex point during a periodic bouncing gait. The flight phase controller is disabled, so that the robot remains un-actuated[4] before it makes ground contact, when high-level controller switching logic (section 4.1.3) enables the neuromuscular stance controller. The simulation is set to end when the following

[3]Because several SEAs share the same drivechain design, only one SEA per design needs to be tuned.

conditions are met:

1. After neuromuscular stance controller is disabled for the first time (i.e. first take-off), wait until it is enabled again (i.e. second touch-down); signal success.
2. When any joint angle exceeds limit; signal failure.
3. When the simulation time exceeds limit (unlikely to ever take-off); signal failure.

In case the simulation succeeds, a finite cost is computed:

$$\text{cost} = \left| \max h_{\text{trunk}} - h_{\text{trunk}}^{\text{apex,des}} \right| \tag{4.29}$$

where $h_{\text{trunk}}^{\text{apex,des}}$ is constant desired apex height. If the simulation fails, the cost is set to a "soft infinity":

$$\text{cost} = C \cdot t_{\text{fail}}^{-1} \tag{4.30}$$

which is lower (better) if the simulation fails later into simulation. This is to provide directional information to guide the optimization to avoid early (likely blatant) failures. Again, CMA-ES is used as the optimization routine. It successfully found a suitable parameter set that enables the stance controller to return the robot back to its starting height (foot 4 cm off ground).

4.2.3 Flight Controller Tuning

With a working stance controller, the flight controller can then be tuned. The tunable parameters are:

1. PD gains for angle and length k_α, d_α, k_l, d_l

[4]Minimum torque limit for keeping cable tension remain (section 4.1.2).

2. Torque cross-compensation terms k_1, k_2
3. Ascent/descent boundary constant k_t

Because changes to the behavior of this controller can be clearly attributed to individual parameters, manual empirical tuning was possible. The following changes are made to the simulation model:

1. Re-enable flight controller (still subject to high-level controller switching)
2. Remove all simulation termination conditions except joint limits, which still signals a failure

Parameters are adjusted after each simulation run according to the behavior. For example, if α regulation results in excessive knee action, k_2 is increased in the next run; if leg length target is held fixed too early, k_t is increased. In around 20 iterations, the controller is already capable of preparing the leg for the next stance well enough.

4.2.4 Stable Bouncing Gait

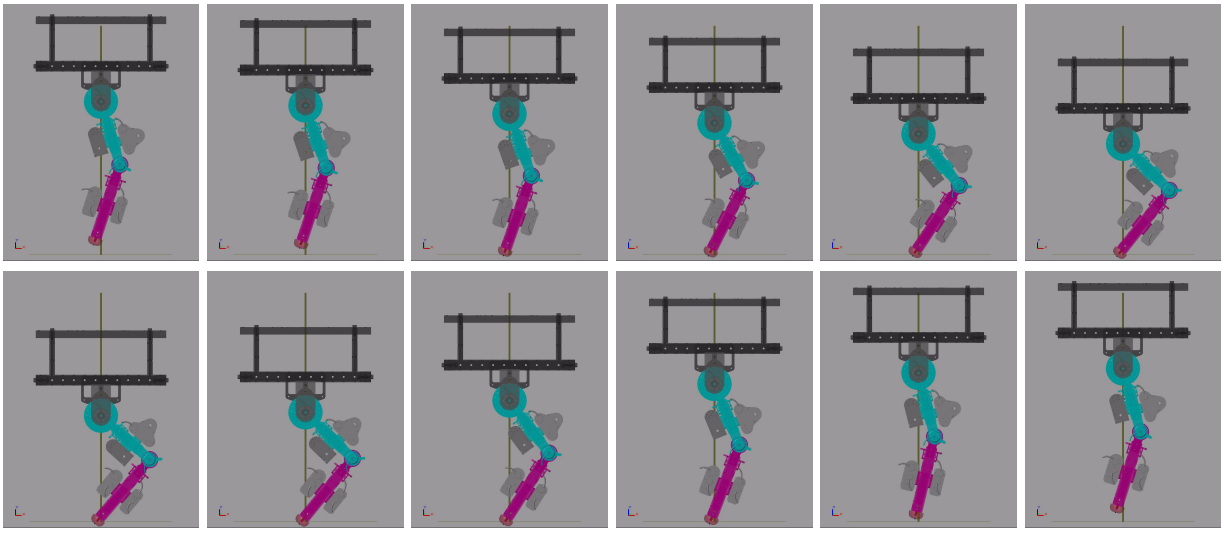


Figure 4.3: Simulation visualization showing one apex-to-apex cycle of periodic bouncing gait; simulation time difference between two adjacent frames: $\Delta t = 1/24\,\mathrm{s}$

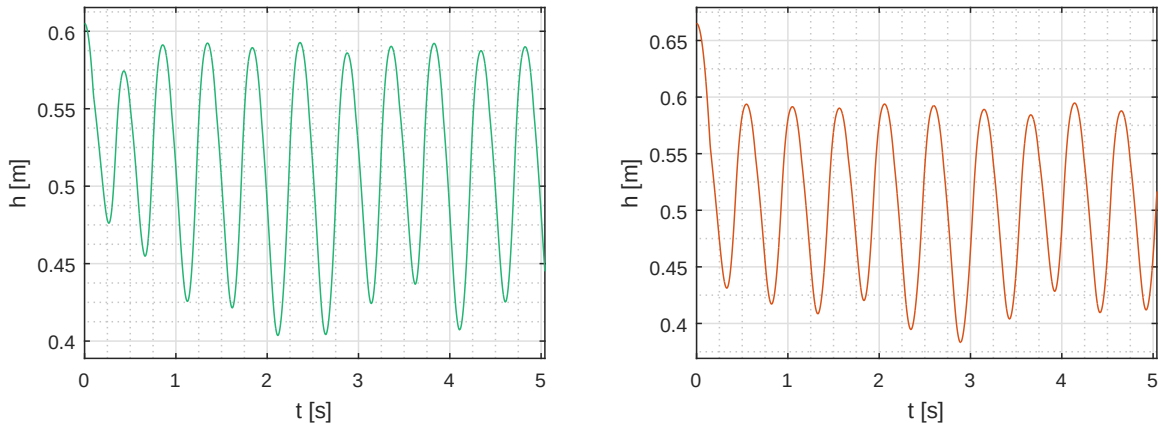


Figure 4.4: Trunk trajectory of two simulated experiments with different initial height

Running the completely tuned simulation model results in a periodic bouncing gait with a stable apex height. Fig. 4.3 shows one full apex-to-apex cycle in this gait, rendered with SimMechanics. The controllers were able to maintain stable apex height regardless of initial height from which the robot was dropped, shown in Fig. 4.4. SEA controllers tracked desired torque profiles well; shown in Fig. 4.5 is the torque tracking of VAS SEA,

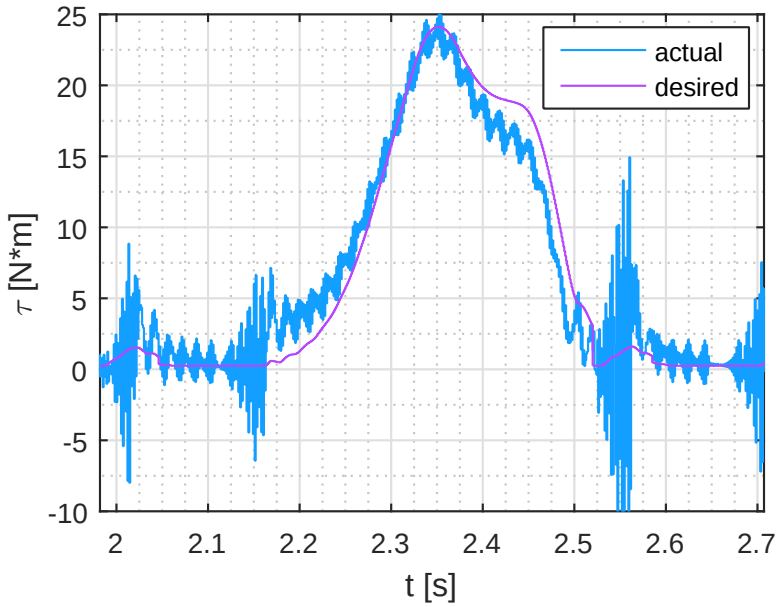


Figure 4.5: VAS SEA torque tracking

which bears the most load. Results from these simulated experiments are further discussed in section 5.1.

Chapter 5

Discussion and Future Work

5.1 Discussion

In section 1.2, two hypotheses are presented:

1. The controller is capable of generating compliant stance leg behavior.
2. The controller is capable of maintaining a stable in-place bouncing gait.

The following discusses whether they are verified or refuted in simulated experiments.

5.1.1 Compliant Leg Behavior

"Compliant leg behavior" refers to the property of human and animal legs that their force-length characteristics resemble that of a potentially nonlinear spring during walking and running gaits. Ideally, during stable in-place bouncing, a compliant robotic leg should also

behave as a spring. There is a quantitative measure of how close a bouncing gait is to that of a perfectly elastic spring called "elasticity coefficient", denoted C_{EL} [8]. It is defined on a plot of ground reaction force (GRF) against compression length of the virtual leg (Fig. 5.1):

$$C_{\mathrm{EL}} = \left(1 - \frac{A}{A_{\max}}\right)^2 \tag{5.1}$$

where $A_{\max} = \max F \cdot \max \Delta l$ and A is the non-directional area enclosed by the curve of one touch-down-to-lift-off cycle[1]. For an ideal spring, regardless of its force-length relationship, $A = 0$ and therefore $C_{\mathrm{EL}} = 1$. For human legs, $C_{\mathrm{EL}} = 0.92 \pm 0.03$ according to measurements on human hopping experiments [8]. This range serves as a criterion for whether a leg shows compliant behavior. The original neuromuscular bouncing stance controller on ideal 2-segment leg simulation can achieve a wide range of C_{EL} as high as 0.95, which satisfies the criterion. On the other hand, data in Fig. 5.1 evaluates to $C_{\mathrm{EL}} = 0.73 \pm 0.06$, which lies far outside the reference range. Therefore, the neuromuscular bouncing stance controller does not yet display compliant leg behavior when implemented on RNL3 simulation.

5.1.2 Stable In-place Bouncing Gait

Several simulated experiments were run with the same parameter set but starting from different initial conditions. All of them were able to maintain a stable in-place bouncing gait as shown in section 4.2.4. While a complete analysis of stability is desirable, existing simulation is sufficient to conclude that the neuromuscular stance controller is capable of maintaining a stable in-place bouncing gait.

[1]Notice this area is different from the directional area $-\int F d\Delta l$ which represents net work done during a stance phase

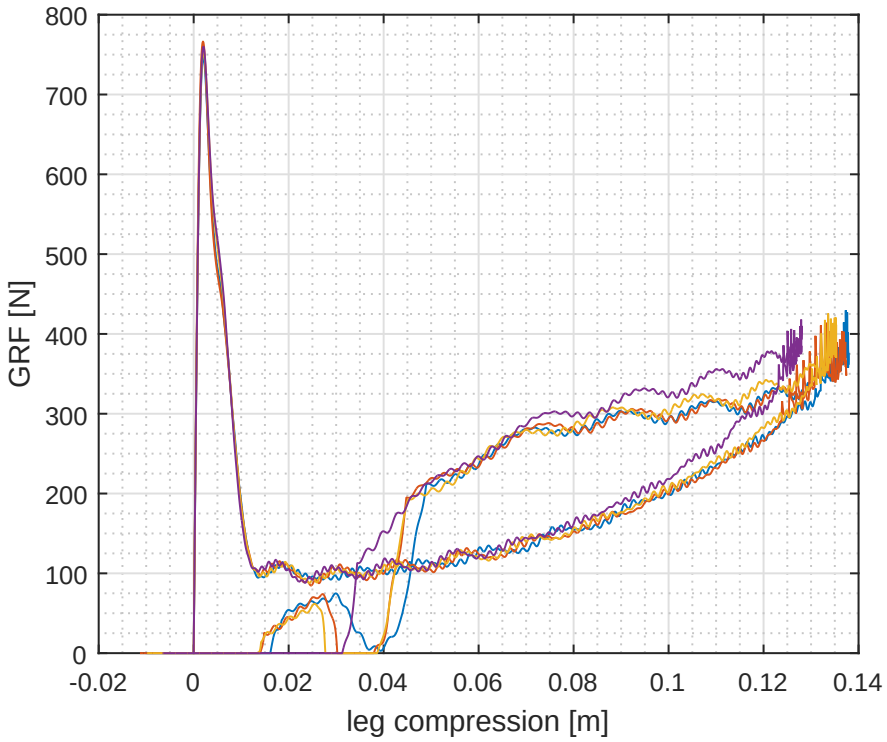


Figure 5.1: Ground reaction force vs. leg compression during a certain cycles in a stable bouncing gait in simulation

5.2 Possible Improvements to RNL3 Hardware

Initial hardware experiments (section 3.4.2) revealed multiple defects in the design of RNL3 testbed. These were not noticed when the neuromuscular swing controller was evaluated on it, likely because the robot bear only the weight of its 2 leg segments, which demanded significantly less load than stance. In order to proceed with hardware experiments, it is necessary to address these defects and improve the performance of hardware. The following discusses each issue and possible solutions.

5.2.1 Testbed Setup

As discussed in section 3.4.1, the original testbed cage design was faulty. While it was intended to provide flexibility of individual control over trunk DoF, its design predates even RNL3 and did not take into account of over-constraining. Although the make-shift boom succeeded as a work-around, it allowed only one approximately vertical DoF. A more practical solution is to employ a traditional sideways-mounted boom like Atrias [**Hereid2014**] in its 2D configuration. Alternatively, current boom configuration can be extended into a Revolute-Prismatic-Revolute (RPR) mechanism, which would allow full planar 3DoF while retaining ability to lock down to only vertical 1DoF.

5.2.2 Actuation

Several actuation cables snapped and were re-attached to the robot. Output drums on SEAs sheared (Fig. 3.18) due to low ultimate tensile strength of UV cure resin (SLA-based 3D printing). While some of them were replaced, they broke again under similar loads. Since the cable and output drums are coupled, a redesign is necessary. The output drum should be made of metal, or directly machined as a part of the output shaft. The new output drum should also accomodate a cable of larger diameter, which allows more safety factor against large loads needed to support the robot's own weight.

A joint roller (see section 3.1.3) yielded under bending load from VAS cable; while it did not break, it also needs to be replaced. This is caused by the overall cable actuation design which exerts significantly higher bending load on two rollers per joint, one on each side, than all other rollers. Since the actuation design will likely remain unchanged, a practical solution is to increase the diameters of these these key rollers so that they can better resist the bending load.

5.2.3 Weight

In section 3.4.1 it was pointed out that the leg is severely overweight. The constant-force spring does not help achieve one of the main design objectives of the hardware platform: that the robotic leg should resemble weight and distribution of a (dynamically scaled) human leg. For the 2-segment and point foot configuration, simply removing the ankle actuators would greatly lighten the severely overweight shank, although this does not apply in the long term due to the eventual addition of actuated ankle-foot (see next section).

5.2.4 Ankle-foot

Currently the leg is missing a fully actuated foot segment. A carbon fiber prosthetic foot of a suitable size is highly desirable. The ankle joint will likely be a scaled-down version of existing hip and knee joints. Rollers might be omitted due to limited room.

5.3 Future Work on Controller Evaluation

Regardless of simulation or hardware, from discussion in section 5.1 it is apparent that there is more work to be done before the neuromuscular stance controller can produce compliant leg behavior. Exact reason for the low C_{EL} is unknown at the moment, but it might be related to how impact and cable actuation influences the dynamics of the leg differently from the ideal model. Stability of the controller, as well as relationship between apex height and key parameters, needs exploring; a poincare return map is an important tool in analyzing these topics.

While current controller assumes point foot and locked trunk, an important direction is to

expand it to more DoF including a 3-segment leg including ankle-foot, and a free 3DoF trunk whose balance must be maintained during stance. As described in section 2.2.3 and [6], bi-articular muscles (and actuators) play an important role in stabilizing such multi-segment mechanisms. A further goal is to establish a mapping from high-level commands such as desired hopping step length and height to controller parameters, achieving truly robust stance leg control that can be combined with the swing leg controller to form a complete bipedal walking system.

Bibliography

[1] Sunil K. Agrawal et al. "Exoskeletons for gait assistance and training of the motor-impaired". In: *2007 IEEE 10th International Conference on Rehabilitation Robotics, ICORR'07* 00.c (2007), pp. 1108–1113. DOI: 10.1109/ICORR.2007.4428562.

[2] Samuel K. Au et al. "Powered ankle-foot prosthesis for the improvement of amputee ambulation". In: *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings* (2007), pp. 3020–3026. ISSN: 05891019. DOI: 10.1109/IEMBS.2007.4352965.

[3] R Desai and H Geyer. "Robust swing leg placement under large disturbances". In: *Proc. IEEE Int Conf on Robotics and Biomimetics*. 2012, pp. 265–270. ISBN: 978-1-4673-2127-3. DOI: 10.1109/ROBIO.2012.6490977.

[4] Ruta Desai and Hartmut Geyer. "Muscle-reflex control of robust swing leg placement". In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2013, pp. 2169–2174. ISBN: 9781467356411. DOI: 10.1109/ICRA.2013.6630868.

[5] Michael F. Eilenberg, Hartmut Geyer, and Hugh Herr. "Control of a Powered Ankle–Foot Prosthesis Based on a Neuromuscular Model". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18.2 (Apr. 2010), pp. 164–173. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2009.2039620. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5378631.

[6] Hartmut Geyer and Hugh Herr. "A Muscle-Reflex Model That Encodes Principles of Legged Mechanics Produces Human Walking Dynamics and Muscle Activities". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18.3 (June 2010), pp. 263–273. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2010.2047592. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5445011.

[7] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. "Compliant leg behaviour explains basic dynamics of walking and running." In: *Proceedings. Biological sciences / The Royal Society* 273.1603 (2006), pp. 2861–2867. ISSN: 0962-8452. DOI: 10.1098/rspb.2006.3637.

[8] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. "Positive force feedback in bouncing gaits?" In: *Proceedings of the Royal Society B: Biological Sciences* 270.1529 (Oct. 2003), pp. 2173–2183. ISSN: 0962-8452. DOI: 10.1098/rspb.2003.2454. URL: http://rspb.royalsocietypublishing.org/cgi/doi/10.1098/rspb.2003.2454.

[9] Robert D. Gregg and Jonathon W. Sensinger. "Towards biomimetic virtual constraint control of a powered prosthetic leg". In: *IEEE Transactions on Control Systems Technology* 22.1 (2014), pp. 246–254. ISSN: 10636536. DOI: 10.1109/TCST.2012.2236840.

[10] Robert D. Gregg et al. "Virtual Constraint Control of a Powered Prosthetic Leg: From Simulation to Experiments With Transfemoral Amputees". In: *IEEE Transactions on Robotics* 30.6 (Dec. 2014), pp. 1455–1471. ISSN: 1552-3098. DOI: 10.1109/TRO.2014.2361937. arXiv: NIHMS150003. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6936867.

[11] Robert D. Gregg et al. "Virtual constraint control of a powered prosthetic leg: From simulation to experiments with transfemoral amputees". In: *IEEE Transactions on*

Robotics 30.6 (2014), pp. 1455–1471. ISSN: 15523098. DOI: 10.1109/TRO.2014.2361937. arXiv: NIHMS150003.

[12] Michael Günther and Hanns Ruder. "Synthesis of two-dimensional human walking: A test of the λ-model". In: *Biological Cybernetics* 89.2 (2003), pp. 89–106. ISSN: 03401200. DOI: 10.1007/s00422-003-0414-x.

[13] Nikolaus Hansen and Andreas Ostermeier. "Completely Derandomized Self-Adaptation in Evolution Strategies". In: *Evolutionary Computation* 9.2 (June 2001), pp. 159–195. ISSN: 1063-6560. DOI: 10.1162/106365601750190398. URL: http://www.mitpressjournals.org/doi/abs/10.1162/106365601750190398.

[14] Archibald Vivian Hill. "The heat of shortening and the dynamic constants of muscle". In: (1978). DOI: http://dx.doi.org/10.1098/rspb.1938.0050.

[15] H. Hultborn and J. B. Nielsen. "Spinal control of locomotion - from cat to man". In: *Acta Physiologica* 189.2 (2007), pp. 111–121. ISSN: 1748-1708. DOI: 10.1111/j.1748-1716.2006.01651.x. URL: http://doi.wiley.com/10.1111/j.1748-1716.2006.01651.x.

[16] Auke Jan Ijspeert. "Central pattern generators for locomotion control in animals and robots: A review". In: *Neural Networks* 21.4 (May 2008), pp. 642–653. ISSN: 08936080. DOI: 10.1016/j.neunet.2008.03.014. URL: http://linkinghub.elsevier.com/retrieve/pii/S0893608008000804.

[17] Ill-Woo Park et al. "Online free walking trajectory generation for biped humanoid robot KHR-3(HUBO)". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* Vol. 2006. May. IEEE, 2006, pp. 1231–1236. ISBN: 0-7803-9505-0. DOI: 10.1109/ROBOT.2006.1641877. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1641877.

[18] Kevin Kemper. *Medulla*. URL: https://code.google.com/p/medulla/ (visited on 12/16/2015).

[19] Brian E. Lawson et al. “Stumble detection and classification for an intelligent transfemoral prosthesis”. In: *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10* (2010), pp. 511–514. ISSN: 1557-170X. DOI: 10.1109/IEMBS.2010.5626021.

[20] A. Schepelmann, J. Austin, and H. Geyer. “Evaluation of Decentralized Reactive Swing-Leg Control on a Powered Robotic Leg”. In: *Proc IEEE Int Conf on Intelligent Robots and Systems, in review*. 2015. ISBN: 9781479999934.

[21] A. Schepelmann, M.D. Taylor, and Hartmut Geyer. “Development of a Testbed for Robotic Neuromuscular Controllers”. In: *Robotics: Science and Systems* (2012). URL: http://www.roboticsproceedings.org/rss08/p49.html.

[22] Seungmoon Song, Ruta Desai, and Hartmut Geyer. “Integration of an adaptive swing control into a neuromuscular human walking model”. In: *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, July 2013, pp. 4915–4918. ISBN: 978-1-4577-0216-7. DOI: 10.1109/EMBC.2013.6610650. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6610650.

[23] A. Seyfarth. “Swing-leg retraction: a simple control model for stable running”. In: *Journal of Experimental Biology* 206.15 (2003), pp. 2547–2555. ISSN: 0022-0949. DOI: 10.1242/jeb.00463. URL: http://jeb.biologists.org/cgi/doi/10.1242/jeb.00463.

[24] A. Seyfarth et al. “Running and Walking with Compliant Legs”. In: *Fast Motions in Biomechanics and Robotics*. Vol. 340. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 383–401. ISBN: 3540361189. DOI: 10.1007/978-3-540-36119-0_18. URL: http://link.springer.com/10.1007/978-3-540-36119-0%7B%5C_%7D18.

[25] Seungmoon Song and Hartmut Geyer. “A neural circuitry that emphasizes spinal feedback generates diverse behaviours of human locomotion”. In: *The Journal of Phys-*

iology 593.16 (Aug. 2015), pp. 3493–3511. ISSN: 00223751. DOI: 10.1113/JP270228. URL: http://doi.wiley.com/10.1113/JP270228.

[26] Seungmoon Song and Hartmut Geyer. "Regulating Speed in a Neuromuscular Human Running Model". In: *15th International Conference on Humanoid Robots (Humanoids)*. 2015, pp. 217–222. ISBN: 9781479968855.

[27] Frank Sup, Amit Bohara, and Michael Goldfarb. "Design and Control of a Powered Transfemoral Prosthesis". In: *The International Journal of Robotics Research* 27.2 (Feb. 2008), pp. 263–273. ISSN: 0278-3649. DOI: 10.1177/0278364907084588. arXiv: NIHMS150003. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2773553%7B%5C&%7Dtool=pmcentrez%7B%5C&%7Drendertype=abstract%20http://ijr.sagepub.com/cgi/doi/10.1177/0278364907084588.

[28] Frank Sup, Huseyin Atakan Varol, and Michael Goldfarb. "Upslope walking with a powered knee and ankle prosthesis: Initial results with an amputee subject". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 19.1 (2011), pp. 71–78. ISSN: 15344320. DOI: 10.1109/TNSRE.2010.2087360.

[29] Frank Sup et al. "Self-contained powered knee and ankle prosthesis: Initial evaluation on a transfemoral amputee". In: *2009 IEEE International Conference on Rehabilitation Robotics, ICORR 2009* (2009), pp. 638–644. ISSN: 1945-7901. DOI: 10.1109/ICORR.2009.5209625. arXiv: arXiv:1011.1669v3.

[30] Michael D Taylor. "A Compact Series Elastic Actuator for Bipedal Robots with Human-Like Dynamic Performance". MA thesis. Pittsburgh, PA: Robotics Institute, Carnegie Mellon University, Aug. 2011.

[31] N Thatte and H Geyer. "Toward Balance Recovery with Leg Prostheses using Neuromuscular Model Control". In: *Biomedical Engineering, IEEE Transactions on* PP.99 (2015), p. 1. ISSN: 0018-9294. DOI: 10.1109/TBME.2015.2472533.

[32] Huseyin Atakan Varol, Frank Sup, and Michael Goldfarb. "Multiclass real-time intent recognition of a powered lower limb prosthesis". In: *IEEE Transactions on Biomedical Engineering* 57.3 (2010), pp. 542–551. ISSN: 00189294. DOI: 10.1109/TBME.2009.2034734. arXiv: NIHMS150003.

[33] Huseyin Atakan Varol, Frank Sup, and Michael Goldfarb. "Powered sit-to-stand and assistive stand-to-sit framework for a powered transfemoral prosthesis". In: *2009 IEEE International Conference on Rehabilitation Robotics, ICORR 2009* (2009), pp. 645–651. ISSN: 1945-7898. DOI: 10.1109/ICORR.2009.5209582. arXiv: NIHMS150003.

[34] Miomir Vukobratovic. "How to Control Artificial Anthropomorphic Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics* 3.5 (Sept. 1973), pp. 497–507. ISSN: 0018-9472. DOI: 10.1109/TSMC.1973.4309277. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4309277.