The Thesis Committee for Rachel May Schlossman

Certifies that this is the approved version of the following Thesis:

Dynamically Consistent Trajectory Planners and Human-Aware Controllers for Human-Centered Robots

APPROVED BY

SUPERVISING COMMITTEE:

Luis Sentis, Supervisor

Efstathios Bakolas

Dynamically Consistent Trajectory Planners and Human-Aware Controllers for Human-Centered Robots

by

Rachel May Schlossman

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2019

Acknowledgments

Thank you so much to my advisor, Luis Sentis, for his fantastic guidance and being a model of passion for robotics. He has consistently made me feel empowered to pursue my ideas. Thank you to Efstathios Bakolas for his openness toward being the second reader of my thesis. Thank you to Ufuk Topcu for his feedback and guidance on my synthesis research. I am very grateful to NASA for their support through the NASA Space Technology Research Fellowship (NSTRF). Thank you to Josh Mehling for being a great mentor and resource at NASA.

One of the things I have most enjoyed about being a member of the Human-Centered Robotics Lab is the amazing generosity of the people in giving each other their time and support. I have also been extremely lucky to make great friends in our lab, with whom I have enjoyed fun meals, basketball games, camping trips, and improv shows. Thanks to Orion Campbell for all of the in-person and phone-based brainstorming sessions, and for a lot of laughs. Thanks to Travis Llado for having interesting commentary about just about any topic, robotics-related or otherwise. Thank you to Gray Thomas for being an incredible research collaborator and mentor. Thank you to Minkyu Kim for being a reliable and fun research collaborator. Thank you to Jaemin Lee for pointing me in helpful directions with his technical insight and for fun conversations. Thanks to Nicolas Brissonneau for his especially positive attitude. Thank you to Junhyeok Ahn for his technical guidance and sense of humor. Thanks to Steven Jorgensen for conveying the importance of patience in debugging,

and for his positive energy and encouragement. Thanks to Mihir Vedantam and Henry Cappel for the fun discussions. Thank you to Kenan Isik for supporting my interest in SEAs. Thank you to Nick Paine for his openness and support as I worked with Apptronik hardware.

Thanks to Liz, Victoria, and Tom for making Austin a wonderful place to live. Thanks to Neal for lots of fun and support. Thank you to my parents. They have always made me feel that any goal I have had is within my reach, and have provided as much support as possible.

My graduate school experience has been incredible and I am grateful to all of these people for their contributions to my academic and personal growth.

Rachel May Schlossman

The University of Texas at Austin
May 2019

Dynamically Consistent Trajectory Planners and Human-Aware Controllers for Human-Centered Robots

Rachel May Schlossman, M.S.E.
The University of Texas at Austin, 2019

Supervisor: Luis Sentis

For robots to successfully be deployed as human assistants in a variety of applications, it is critical that the robots' controllers and planners are designed with the considerations of both the robots' and humans' abilities and needs. In space applications, where energy is a finite and limiting resource in missions, it may prove necessary to exploit the energy storing-component of series elastic actuators to meet the efficiency needs, while operating in harsh and varied environments. In human-occupied workplaces, robots can only provide the needed support to humans if the robot controller can properly reason about and react to humans' requirements and capabilities. This thesis presents and assesses strategies to address these kinds of scenarios.

In Chapter 2, we present a trajectory optimization scheme based on sequen-

tial linear programming to leverage the energy-storing capabilities of series elastic actuators for high-performance tasks. We discuss the current limitations in optimization strategies for series elastic actuated robots. One of the difficulties of this planning problem is respecting all relevant, low-level actuator constraints and handling system nonlinearities in a computationally efficient manner. Our simulation and hardware experiments demonstrate the leveraging of compliance for faster motions as compared to those that are achieved by the compliant systems' rigid counterparts.

Chapter 3 addresses the need for reactive synthesis to be employed to automatically devise human-aware robot controllers for scenarios in which humans and robots continuously collaborate. Through this approach, it is possible to synthesize high-level control policies that are formally guaranteed to meet human requirements. We present a case study in which a robot seeks to deliver work to a human so that the human is productive, but not stressed by her work backlog. We demonstrate the achievement of a human productivity-informed controller using a mobile manipulator robot that picks up and delivers work based on work backlog. One of the challenges of this problem is devising human productivity models that are practical and accurate. We explore a toy scenario in the hope that this research will introduce methodologies that can be generalized for more practical cases.

Contents

Acknowledgments **iv**

Abstract **vi**

List of Tables **xi**

List of Figures **xii**

Chapter 1 Introduction **1**

1.0.1 Thesis Contributions 2

Chapter 2 Exploiting the Natural Dynamics of Series Elastic Robots by Actuator-Centered Sequential Linear Programming **4**

2.1 Introduction 4

2.1.1 Related Works 5

2.1.2 Sequential Linear Optimization 6

2.1.3 Overview 7

2.2 Modeling 8

2.2.1 Actuator Dynamics 8

2.2.2 Robot Dynamics 11

2.2.3 Discretization 12

2.3 Iterative Linear Programming 13

2.3.1 Algorithm Tuning and Robustness . . . 15
2.4 Simulation . . . 16
2.4.1 Apptronik^{TM} Draco-Inspired System . . . 16
2.4.2 Ground Contacts . . . 18
2.4.3 Velocity Maximization for Jumping . . . 19
2.4.4 Zero Input Behavior . . . 23
2.4.5 Pseudo-Mass Selection . . . 24
2.5 Algorithm Tuning . . . 26
2.6 Experiments . . . 27
2.7 Discussion . . . 31

Chapter 3 Toward Achieving Formal Guarantees for Human-Aware Controllers in Human-Robot Interactions 33

3.1 Introduction . . . 33
3.2 Preliminaries . . . 35
3.3 Work Delivery with Human Backlog Model . . . 37
3.3.1 Modeling . . . 37
3.3.2 Reactive Synthesis . . . 41
3.3.3 Controller Synthesis . . . 46
3.4 Experiments . . . 47
3.4.1 Toyota Human Support Robot . . . 47
3.4.2 Controller Implementation . . . 48
3.4.3 Results . . . 49
3.5 Discussion . . . 53

Chapter 4 Conclusions 54

Appendix A Equilibrium Nominal Trajectory 56

Bibliography **58**

List of Tables

Table 2.1 Dynamics 18
Table 2.2 Transmissions (Fig. 2.2.b) 18
Table 2.3 Constraints 18
Table 2.4 Average Time per Iteration for Algorithm Components and Total Simulation Time (s) 23
Table 2.5 P170 Identified Parameters 28
Table 2.6 P170 Constraints 28
Table 2.7 Experiment Results 31

List of Figures

Figure 2.1 Internal dynamics of the SEA for the three-mass, differential constraint model. While there are no fluids in the physical SEA system, a fluid differential is used as a metaphor for the real mechanical differential, to easily visualize that the back forces are equal and that the motions of the spring and motor subsystems are in series. A pseudo-mass term, M_p, is introduced to allow discretization with longer time steps. 9

Figure 2.2 (a) Draco leg prototype. Our simulation-only experiments are modeled after this robot, with significantly reduced spring rates, performing the liftoff phase of a jump. (b) Schematics emphasize the nonlinear transmissions between actuator length, z, and joint angle, q, for the ankle and knee. These nonlinear transmissions motivate our choice to represent the robot impedance in actuator-length–actuator-force space rather than the standard joint-angle–joint-torque space. . 17

Figure 2.3 The simulated robot with point contacts at the front (left) and back (right) of the foot. 19

Figure 2.4 With the 25th iteration trajectory from the compliant jumping study as a baseline, the results suggest that the error decays exponentially. 20

Figure 2.5 (a) Spring deflection trajectories for the compliant leg's optimal behavior. (b) The corresponding optimal u's to produce the optimal trajectory, which operate at the input limits. (c) The z trajectories produced over 20 iterations to produce the jumping behavior, demonstrating convergence. 21

Figure 2.6 The rigid robot, left, and compliant robot, right (with the springs indicated in pink), after they jump and return to the ground. The COMs of the two robots are illustrated as black triangles. The two COM initial heights are both 0.67 m when the robots lift into the air, and the maximum heights of the compliant and rigid configuration COMs are 0.86 m and 0.81 m, respectively, which are marked with red lines. 22

Figure 2.7 This comparison between the upward velocity components of the optimal trajectories for the rigid and compliant systems shows that the final velocity of the compliant system is 1.2 times that of the rigid system. 23

Figure 2.8 The z trajectories produced over 15 iterations (M_p = 580 kg and ΔT = .0095 s), showing convergence for the system's zero input behavior. 24

Figure 2.9 The natural frequency of A_1 (linear time-invariant actuator dynamics with pseudo-mass) varies with M_p, and we select an M_p value where the frequency aligns with that of the nonlinear, continuous dynamics at 35 rad/s. 25

Figure 2.10 Our algorithm converges as α is tuned, demonstrating the limits on optimal upward COM velocity, V^*, as the penalty increases. 26

Figure 2.11 The Taurus testbed, with an SEA whose spring is softer than Draco's viscoelastic elements by an order of magnitude. For our experiment, a 5 lb weight is attached to the actuator arm. 27

Figure 2.12 Tuning pseudo-mass. (a) Maximum eigenvalue frequency for true arm-actuator system (Continuous) and various approximations (A_1) for the actuator alone with varying pseudo-mass (M_p). The arm points down at 0 rad. Changing eigenvalue frequency for the true system is due to angle-dependent reflected inertia. The choice of 220 kg is relatively accurate in the operational region centered around 1.57 rad. (b) Simulation error of discrete-time models used for trajectory optimization as a function of pseudo-mass, for a feedforward trajectory in the operational region, with a fixed time step. Average squared error relative to the trajectory of the true model. The low pseudo-mass example has aliasing errors. High pseudo-mass errors exist, but are not as extreme. 29

Figure 2.13 (a)-(e) show the simulated (red) and actual (green) optimal behavior of the P170 actuator when spring dynamics are considered. The high-quality tracking in (a)-(e) support that the system has been well identified. (f)-(j) show the expected and actual behaviors of the actuator when spring dynamics are not considered. These results demonstrate dynamic inconsistencies in tracking when the spring subsystem is neglected in planning. The experimental data are shifted by 5 ms to account for time delay. Fifteen seconds are used to interpolate to the starting position, and the optimal motion begins at 15 s. 30

Figure 3.1 (a) The Toyota Human Support Robot, which we use as our experimental platform for human-informed work delivery, picking up completed work. (b) The HSR dropping off completed work at the Inventory Station. 36

Figure 3.2 Human Model with three states. Transitions are triggered by guards, g, and there are corresponding outputs, y. In this transition system, edges are labeled in a g / y format. In this system, the outputs are described by v_2, v_3, and v_4. 39

Figure 3.3 Robot Model with N = 3. State 0 is the Inventory Station and State 3 is the Human Workstation. Each edge of the TS is labeled with a guard and an action. The presence of an obstruction in one of the robot's adjacent positions can restrict the robot's next action. The guards express whether or not there is an obstacle blocking the robot from proceeding to a neighboring state, and we define $g_1 \triangleq \mathcal{O}_1$ and $g_2 \triangleq \mathcal{O}_2$ 40

Figure 3.4 HRI Work Delivery and Pickup Transition System with N = 3. The edges of the transition system are labeled first by guards (g_3, g_4), then by actions, and lastly by pertinent output variables (v_1, v_2, v_3, v_4). The guards which determine whether the human is working or waiting are expressed by $g_3 \triangleq \bigcirc BL > 0\%$ and $g_4 \triangleq \bigcirc BL = 0\%$. As illustrated in Fig. 3.3, the robot cannot move to a neighboring state if it contains an obstacle, but we do not show this guard due to space limitations. 43

Figure 3.5 The communication protocols between the the robot and the sub-task (SMACH) and high-level (Slugs) controllers. The systems communicate by repeating (a)-(h), where: (a) Track and command sequential tasks; (b) Perceive, navigate, and manipulate; (c) Sequential tasks complete; (d) Request next action; (e) Subscribe to ROS environment topics; (f) Check for human at workstation. Update RS and BL when worker is present; (g) Select next action from look-up table. (h) Respond with next action. 49

Figure 3.6 Experimental setup with four positions in the workspace, corresponding to $RS = 0$ (Inventory Station), 1, 2, and 3 (Human Workstation). The HSR transports deliverables in its satchel and carries completed work in its manipulator back to the Inventory Station. A human obstructs its path, causing it to remain in State 2 until the next time step, by which time the human will have departed. 50

Figure 3.7 Robustness test with $BL_{init} = 40\%$. Human obstacles that appear in States 1 and 2 and depart by the next time step are marked by black x's. The worker moves out of the workspace for three minutes, which is highlighted in pink. The robot waits for the human to return at $RS = 2$, but does not update RS and the environment variables in the Slugs planner until the human returns to work. . . . 51

Figure 3.8 Robot behavior with $BL_{init} = 30\%$. When $RS = 1$ and HF at 1.75 minutes, the SMACH sub-task controller tracks the amount of time the robot takes to travel from State 1 to State 0 and execute the dropoff sub-tasks, during which time, tries = 1. If this time exceeds 35 seconds, $\neg\mathcal{S}$ is communicated to the high-level controller. The Slugs planner updates the tries value to be equal to 2, and commands that the robot continue to execute its dropoff sub-tasks in the next time step. After the second try, the robot successfully drops off the work, and the robot's manipulator is empty again. 52

Figure 3.9 Robot behavior with $BL_{init} = 86.7\%$. The circular markers indicate the times at which the sub-task controller calls the high-level decision tree to request the next robot action. Excluding its initial movement to and from State 1, the robot elects to wait at State 0 until it moves to deliver work, even when there are no obstacles present. 52

Chapter 1

Introduction

The potential benefits of future robotic assistants and caretakers are unbounded. However, these gains can only be maximized through low-level and high-level planners and controllers that are aligned with their desired purposes, from duties like strenuous, manual tasks to delivering work to humans at appropriate times. This thesis explores two facets of robot operation: (1) generating optimal trajectories to exploit the robot's low-level, energy storing components, and (2) automatically synthesizing high-level robot controllers that are formally guaranteed to meet a human's productivity needs.

In Chapter 2, we consider that series elastic robots are best able to follow trajectories that obey the limitations of their actuators, since they cannot instantly change their joint forces. In fact, the performance of series elastic actuators can surpass that of ideal force source actuators by storing and releasing energy. In this chapter, we formulate the trajectory optimization problem for series elastic robots in a novel way, based on sequential linear programming. Our framework is unique in the separation of the actuator dynamics from the rest of the dynamics, and in the use of a tunable pseudo-mass parameter that improves the discretization accuracy of our approach. The actuator dynamics are truly linear, which allows them to be

excluded from trust-region mechanics. This causes our algorithm to have similar run times with and without the actuator dynamics. We demonstrate our optimization algorithm by tuning high performance behaviors for a single-leg robot in simulation and on hardware for a single degree-of-freedom actuator testbed. The results show that compliance allows for faster motions and takes a similar amount of computation time.

Chapter 3 focuses on the human-centered ingredient of robot operation. With the primary objective of human-robot interaction being to support humans' goals, there exists a need to formally synthesize robot controllers that can provide the desired service. Synthesis techniques have the benefit of providing formal guarantees to meet specifications. There is potential to apply these techniques for devising robot controllers whose specifications are coupled with human needs. This chapter explores the use of formal methods to construct human-aware robot controllers to support the productivity requirements of humans. We tackle these types of scenarios via human workload-informed models and reactive synthesis. This strategy allows us to synthesize controllers that fulfill formal specifications that are expressed as linear temporal logic formulas. We present a case study in which we reason about a work delivery and pickup task such that the robot increases worker productivity, but not stress induced by high work backlog. We demonstrate our controller using the Toyota Human Support Robot (HSR), a mobile manipulator robot. The results demonstrate the realization of a robust robot controller that is guaranteed to properly reason and react in collaborative tasks with human partners.

1.0.1 Thesis Contributions

The contributions of this thesis are twofold:

1. A computationally efficient trajectory optimization scheme that respects all low-level state and input constraints while leveraging the natural dynamics of

series elastic actuators.

2. An investigation on employing reactive synthesis to devise a human-aware robot controller that supports a human and robot's collaborative work.

Through exploring low-level, mechanical and high-level, behavioral considerations, we make strides toward the ultimate goal of devising robots that fulfill human desires and needs.

Chapter 2

Exploiting the Natural Dynamics of Series Elastic Robots by Actuator-Centered Sequential Linear Programming

2.1 Introduction

Since its inception [1], a primary drawback of series elastic actuation has been the additional challenge for the control system. Human-centered robots commonly make use of series elastic actuators (SEAs), which offer the benefits of compliance—for safe interaction with humans—increased robustness, and force sensing [2]. The compliant element is able to store and release energy, like human muscles, presenting an opportunity for increased efficiency and agility as compared to rigid actuators [3].[1] Both feedback controllers and trajectory planners are faced with a more com-

[1]This chapter contains material from [4]. My main contributions include simulation development, system identification execution, and hardware implementation and experimentation.

plex challenge when interfacing with these systems, yet modern control systems for human-centered robots (e.g., [5]) rely on a force-control planning abstraction which specifies an unmeetable goal for the low level feedback controller and provides those controllers with planned trajectories that do not respect their dynamic limitations. Our work addresses some of these issues.

2.1.1 Related Works

Interest in modified series elastic actuators with clutches and variable stiffness compliant elements has driven many groups to derive bang-bang style and cyclic optimal behaviors to illustrate improved mechanical performance [6]. Few groups, however, have investigated more general behaviors that allow for nonlinearities in the system. In [7], a convex optimization problem is formulated to maximize joint velocity by computing the switching times between rigid and compliant actuator behavior via the use of a clutch, but the actuator dynamics are linear except at switching times. One of the contributions of our work is the ability to handle the nonlinearities that are introduced at all time steps through a nonlinear transmission, while still leveraging compliance.

Iterative regulator-based optimal control has been successful in handling nonlinearities in these systems and achieving rapid motions in compliant robots, but is restricted in capturing state and input constraints, e.g., transmission speed or spring deflection limits. The iLQR indirect method has been modified to allow input constraints [8], [9]—but not state constraints directly. In [9], iLQR is used in combination with variable stiffness actuators to leverage the energy storing capability of the compliant element to throw a ball, but the motor position constraint can only be captured indirectly through the input constraint. Inequality state constraints in [10] are reformulated as canonical input constraints, yet the number of constraints possible with this strategy is at most the number of inputs. In contrast,

our work captures all linear state and input constraints, which are upheld by the linear program.

A review of trajectory optimization techniques for series elastic walking provides insight into additional strategies that allow compliance while upholding constraints. Hybrid zero dynamics strategies have employed virtual, holonomic, and actuator constraints while implementing nonlinear programming (NLP) [11], [12] to minimize the energy consumption during each step. Another strategy, employed in the COMAN robot, mimicked human center of mass trajectories and tuned stepping frequency to best use the compliant elements [13]. Our work offers the benefit of exploring problem structure at a lower level to efficiently handle nonlinearities and leverage compliance. We do so via sequential linear optimization, which allows us to leverage the speed of solving linear subproblems.

Spline-parameterized, nonlinear programming (NLP) and collocation approaches, based on general purpose large-scale NLP libraries, have been successfully applied to series elastic robots. In [14], optimal walking trajectories are produced via NLP to be consistent with compliant dynamics subject to all relevant constraints with pre-defined contact transition times. [15] adds a collocation method to automatically select contacts, to automatically generate multiple steps of walking, and to jump, at the cost of approximating some actuator constraints. This approach leverages powerful and highly general NLP libraries, however, these general solvers result in long run-times on the order of an hour, even for problems that have roughly the same number of trajectory parameters as ours[2].

2.1.2 Sequential Linear Optimization

Linear optimization allows for the minimization of a linear cost function, $h(x)$, subject to m linear inequality constraints, $g(x)$, and n linear equality con-

[2] 1,782 parameters in "less than an hour" [15] versus our 1,176 parameters in 28.5 seconds for a two-link leg—iterating an LP 19 times.

straints, $r(x)$, related to the states and input constraints:

$$\begin{aligned} &\underset{x}{\text{minimize}} && h(x) \\ &\text{subject to} && g_i(x) \leq b_i,\ i = 1, \ldots, m. \\ & && r_i(x) = c_i,\ i = 1, \ldots, n \end{aligned}$$

where $h(x)$, $g(x)$, $r(x)$, b, and c are scalars [16], [17]. As an extension, sequential linear optimization allows for local optimization problems to be solved such that the nonlinearities in a system can be considered [17]. A linear program can be used to perform optimization on our nonlinear robot model by formulating a robot model that is locally linear over a single time step. Sequential linear optimization can then iteratively develop a trajectory for the time-varying linear system. The strategy also offers the advantage of upholding all state and input constraints, which indirect optimization methods can only approximate.

2.1.3 Overview

In this chapter, we propose a direct optimization algorithm which efficiently considers the nonlinear effects of the transmission linkages, robot dynamics, input and state constraints, and the energy storing capabilities of the series elastic elements. The algorithm uses sequential linear optimization to minimize a final velocity objective (with a 1-norm input penalty) to demonstrate its ability to produce high performance behaviors while satisfying system constraints. We formulate the problem as input selection for a time-varying discrete time linear system approximation that is updated iteratively. We formulate the system dynamics to connect the actuator space to the joint space. One of the key, novel features of our approach is the use of a fictitious pseudo-mass to improve discretization accuracy for the actuator component at large time steps. We find the pseudo-mass' value must be close to the reflected robot inertia to minimize simulation error and eigenvalue approximation

error. A pseudo-mass of 0 kg results in unacceptable discretization inaccuracy. By exploiting problem structure via separating the linear and nonlinear components of our model, we typically achieve convergence within 20 iterations for a two-link system. Convergence is achieved more quickly when we test our approach on hardware for a single degree-of-freedom testbed. Our experiments demonstrate a greater degree of dynamic consistency and the leveraging of compliance when the spring dynamics are considered for trajectory generation.

2.2 Modeling

2.2.1 Actuator Dynamics

Our model considers internal actuator dynamics, which are common for control design, but rare for trajectory design due to computational complexity. We follow the advice of [18] and [19], and connect three second-order systems through a differential to develop an unlumped model of the SEA.

The actuator model, shown in Fig. 2.1, comprises the spring system; the motor system with input current, u; and the load system. The states considered are spring displacement, δ; spring velocity, $\dot{\delta}$; motor displacement, y; and motor velocity, $\dot{y}$. The variables z and $\dot{z}$ correspond to total actuator length and velocity, respectively. The motor subsystem is reflected to prismatic motion through the transmission—hence, all parameters of the subsystems are in linear units. The three systems are connected through a three-way mechanical differential, $\mathcal{D}$, which enforces the relationship:

$$z = \delta + y. \tag{2.1}$$

The dynamics of the three subsystems are:

$$M_s\ddot{\delta} + \beta_s\dot{\delta} + k\delta = -f, \tag{2.2}$$

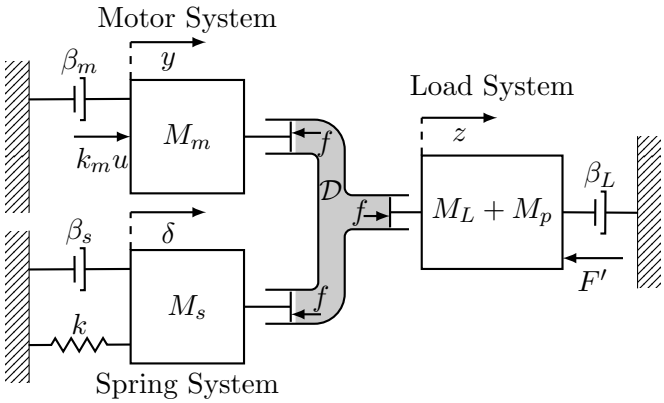

Figure 2.1: Internal dynamics of the SEA for the three-mass, differential constraint model. While there are no fluids in the physical SEA system, a fluid differential is used as a metaphor for the real mechanical differential, to easily visualize that the back forces are equal and that the motions of the spring and motor subsystems are in series. A pseudo-mass term, M_p, is introduced to allow discretization with longer time steps.

$$(M_L + M_p)\ddot{z} + \beta_L \dot{z} = f - (F - M_p \ddot{z}), \tag{2.3}$$

$$M_m \ddot{y} + \beta_m \dot{y} = k_m u - f. \tag{2.4}$$

M_s, M_L, and M_m are the masses of the spring, load, and motor systems, respectively; β_s, β_L, and β_m are these systems' respective damping coefficients; k is the spring constant; and k_m is the reflected motor constant. The second input, F, is the force output from the actuator, which is used to link with the robot dynamics and the nonlinearities in the system. M_p is a fictitious pseudo-mass, which will be used to tune the eigenvalues of the linear actuator system before discretization, as discussed in Section 2.3. We define F' as:

$$F' \triangleq F - M_p \ddot{z}. \tag{2.5}$$

The variable f is equal to the back forces from the differential and, equivalently, the Lagrange multiplier which enforces the differential constraint. Substitution for f reveals that this model is ultimately fourth order:

$$E_o \dot{x} = A_o x + B_{o,u} u + B_{o,F} F', \tag{2.6}$$

where state vector $x \triangleq \begin{bmatrix} \delta & \dot{\delta} & y & \dot{y} \end{bmatrix}^T$ and

$$E_o \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & M_s + M_L + M_p & 0 & M_L + M_p \\ 0 & 0 & 1 & 0 \\ 0 & M_L + M_p & 0 & M_m + M_L + M_p \end{bmatrix}$$

$$A_o \triangleq \begin{bmatrix} 0 & 1 & 0 & 0 \\ -k & -(\beta_s + \beta_L) & 0 & -\beta_L \\ 0 & 0 & 0 & 1 \\ 0 & -\beta_L & 0 & -(\beta_L + \beta_m) \end{bmatrix}$$

$$B_{o,u} \triangleq \begin{bmatrix} 0 & 0 & 0 & k_m \end{bmatrix}^T, \quad B_{o,F} \triangleq \begin{bmatrix} 0 & -1 & 0 & -1 \end{bmatrix}^T.$$

Rearranging (2.6),

$$\dot{x} = A_* x + B_{*,u} u + B_{*,F} F', \tag{2.7}$$

where

$$A_* \triangleq E_0^{-1} A_0, \quad B_{*,u} \triangleq E_0^{-1} B_{o,u}, \quad \text{and}$$

$$B_{*,F} \triangleq E_0^{-1} B_{0,F}.$$

From the construction of A_0, it is clear that the eigenvalues of A_* will vary with M_p. As we proceed, we will discuss the application of this formulation for the general case of p joints. Our state vector will be extended to:

$$x = [x_1^T, x_2^T, \ldots, x_p^T]^T, \tag{2.8}$$

where each x_i captures the four states described in (2.6) for their respective actuator system. Equation (2.6) is extended (using the Kronecker product $\otimes$) to a p-link system with:

$$E_{o,p} = I_p \otimes E_o, \quad A_{o,p} = I_p \otimes A_o, \tag{2.9}$$

$$B_{o,u,p} = I_p \otimes B_{o,u}, \quad \text{and} \quad B_{o,F,p} = I_p \otimes B_{o,F}, \tag{2.10}$$

where I_p is the pxp identity matrix. Equation (2.7) can then be reformulated using (2.9) and (2.10) to obtain A_1, $B_{1,u}$, and $B_{1,F}$ for p joints:

$$\dot{x} = A_1 x + B_{1,u} u + B_{1,F} F'. \tag{2.11}$$

2.2.2 Robot Dynamics

The force F connects the actuator to the robot dynamics. In general, for a multi-link system, the dynamics are:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau = L(q)^T F, \tag{2.12}$$

where M, C, and G represent inertia, Coriolis and centrifugal, and gravitational forces, respectively, and q is the generalized joint angle vector.

The angle-dependent moment arm between the actuator and the joint, $L(q)$ abbreviated L, serves as the Jacobian between the joint space and the actuator space: $L\dot{q} = \dot{z}$. We solve for F' by projecting it into the actuator-position–actuator-force space and manipulating (2.12):

$$F' = F - M_p \ddot{z} = (L^{-T} M(q) L^{-1} - M_p)\ddot{z} + b(q,\dot{q}), \tag{2.13}$$

where

$$b(q,\dot{q}) \triangleq L^{-T}(C(q,\dot{q}) + G(q) - M(q)L^{-1}\dot{L}\dot{q}). \tag{2.14}$$

This is an expression for the impedance of the robot at the $\{\dot{z}, F'\}$ port.

2.2.3 Discretization

To prepare for discrete time u optimization, the state space model is discretized into N time steps of length ΔT. By the continuous state space model in (2.11), acceleration at the actuator output can be computed as:

$$\ddot{z} = S(A_1 x + B_1 \begin{bmatrix} u \\ F' \end{bmatrix}), \tag{2.15}$$

where B_1 is the concatenation of $B_{1,u}$ and $B_{1,F}$. This is actuator admittance at the $\{\dot{z}, F'\}$ port. S is formulated to capture the acceleration terms for the p-link system:

$$S = I_p \otimes \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}. \tag{2.16}$$

F' is expressed in terms of the states by substituting (2.15) into (2.13):

$$\begin{aligned} F' =& [I - (L^{-T} M(q) L^{-1} - M_p) S B_{1,F}]^{-1} [b(q, \dot{q}) + \\ &+ (L^{-T} M(q) L^{-1} - M_p)(S A_1 x + S B_{1,u} u)]. \end{aligned} \tag{2.17}$$

We discretize the linear actuator admittance model under the zero-order hold assumption for both u and F'. The discrete state space model is then:

$$x_{n+1} = A x_n + B \begin{bmatrix} u_n \\ F'_n \end{bmatrix}, \tag{2.18}$$

where

$$A \triangleq e^{A_1 \Delta T}, \quad B \triangleq \int_0^{\Delta T} e^{A_1 (\Delta T - \tau)} B_1 d\tau. \tag{2.19}$$

We combine discrete time admittance and impedance at the $\{\dot{z}, F'\}$ interface by grouping terms which are linear in x and u. The discretized (time-varying)

update equation is:

$$x_{n+1} = A_{lin,n}x_n + B_{lin,n}u_n + bias_n, \tag{2.20}$$

where A_{lin} and B_{lin} capture the linear dynamics associated with the actuator states and input current, respectively, and $bias$ captures the nonlinear robot impedance, including gravity, Coriolis effects, and nonlinear transmissions. The M_p parameter is used to minimize the error introduced by discretizing the actuator admittance in the absence of the reflected inertia of the robot links. The x and u vectors at each time step are concatenated to form the trajectory matrices $\mathbf{X} \triangleq [x_1, x_2, ..., x_N]$ and $\mathbf{U} \triangleq [u_1, u_2, ..., u_{N-1}]$, respectively. This locally-linear model forms the foundation from which our algorithm is developed.

2.3 Iterative Linear Programming

For trajectory optimization of a p-link system, our approach follows a strategy that culminates in a linear programming subproblem. Our local optimization approach requires a baseline trajectory, $\mathbf{Z}_{base}$, the concatenation of z_{base} over all time steps, to initialize the nonlinear parts of the dynamics. A slow trajectory or a static position both serve as good choices. There is no need for a similar baseline trajectory for the actuator states due to our exploitation of their linear problem structure. The $\mathbf{Z}_{\text{base}}$ trajectory allows us to compute the time-varying matrices used in (2.17) to compute F'. We can then compute the linearization components, A_{lin}, B_{lin}, and $bias$, for each time step (effectively saving our solver from eliminating the F' variable itself).

The linear problem structure can then be exploited. New displacement and velocity trajectories for the spring and motor subsystems are computed via a linear program and are captured in the optimal trajectory, $\mathbf{X}^*$. The optimal control parameters over all time steps, captured in $\mathbf{U}^*$, are also produced. The resulting

$\mathbf{Z}$ trajectory becomes the new $\mathbf{Z}_{\text{base}}$, and $\mathbf{X}^*$ is used to compute the new F', A_{lin}, B_{lin}, and *bias* matrices for the next iteration. Trust region constraints will keep the next $\mathbf{Z}$ trajectory close to this updated $\mathbf{Z}_{\text{base}}$ trajectory. The algorithm continues to run until the 2-norm of the difference between the current and previous trajectories stops changing.

A key benefit of our approach is that all relevant actuator state and input constraints can be included in the formulation. The constraints are associated with the upper and lower bounds of the allowable spring deflections, $\overline{\delta}$, joint limits, actuator ballscrew velocity, $\overline{\dot{y}}$, and input currents, $\overline{u}$. The parameter $\overline{\Delta z}$ defines the trust region, which can be used to aid convergence of the iteration scheme. We note that the dimension of this trust region is small relative to the full dimension of $\mathbf{X}$—again due to separation of the linear and nonlinear dynamics. The final state can be subject to partial end point constraints. Our linear subproblem minimizes a problem-specific, linear cost function, $h(\mathbf{X}, \mathbf{U})$, which is a function of, and is subject to linear constraints on, the discretized states and inputs:

$$
\begin{aligned}
&\underset{\mathbf{X},\mathbf{U},\mathbf{U}_{abs}}{\text{minimize}} && h(\mathbf{X}, \mathbf{U}) && && (2.21)\\
&\text{subject to} && \text{dynamics: (2.20)} && \forall\ n \in \mathcal{N}/N &&\\
&\text{a trust region:} &&&&&&\\
& && |z_{i,n} - z_{i,n,base}| \leq \overline{\Delta z} && \forall\ i \in \mathcal{P},\ n \in \mathcal{N} &&\\
&\text{state and input constraints:} &&&&&&\\
& && |\delta_{i,n}| \leq \overline{\delta} && \forall\ i \in \mathcal{P},\ n \in \mathcal{N} &&\\
& && z_{min,i} \leq z_{i,n} \leq z_{max,i} && \forall\ i \in \mathcal{P},\ n \in \mathcal{N} &&\\
& && |\dot{y}_{i,n}| \leq \overline{\dot{y}} && \forall\ i \in \mathcal{P},\ n \in \mathcal{N} &&\\
& && |u_{i,n}| \leq \overline{u} && \forall\ i \in \mathcal{P},\ n \in \mathcal{N}/N &&
\end{aligned}
$$

and problem-specific constraints, in two our studies:

$$x_1 = x_{\text{init.}},\ z_N = z_{\text{fin.}}$$
$$|u_{dev,i,n}| \leq u_{abs,i,n} \qquad \forall\ i \in \mathcal{P},\ n \in \mathcal{N}/N$$

and in our single-leg simulation only:

$$J_{\text{com\_x\_velocity}} \dot{z}_N = 0$$
$$\Phi_{i,n} \geq 0 \qquad \forall\ i \in 4,\ n \in \mathcal{N}/N$$

where $|a| \leq b$ is shorthand for two linear inequalities, $-b \leq a \leq b$; $\forall\ n \in \mathcal{N}$ means $n = 1, \ldots, N$; $\forall\ i \in \mathcal{P}$ means $i = 1, \ldots, p$; and / means omitting an element from the set. The parameter Φ refers to the foot contact constraints, which will be discussed in Section 2.4.2. The variables u_{dev} and u_{abs} are used to minimize deviations from an equilibrium input trajectory, as discussed in Section 2.4.3. The specific cost functions used in our studies and problem-specific constraints are described further in the Simulation and Experiments sections.

2.3.1 Algorithm Tuning and Robustness

To achieve convergence, we choose $\overline{\Delta z}$ to limit planning to the region where our linearized dynamics are not too inaccurate. Our novel approach is to select M_p so that the fastest eigenvalue over the entire trajectory (which corresponds to the spring oscillation mode in the systems we studied) of A_{lin} and A_1 approximates the fastest eigenvalue of the continuous system, ensuring an accurate approximation of the system dynamics. When $M_p = 0$, the spring dynamics settle faster than one time step and cannot be leveraged.

Each iteration of our algorithm relies on the optimal trajectory from the previous iteration, $\mathbf{X}^*_{j-1}$, to generate F', A_{lin}, B_{lin}, $bias$, and $\mathbf{X}^*_j$. Hence, without attention to robustness, an infeasible solution will lead to algorithmic failure. We use Algorithm 1 to resolve this issue. Essentially, if the program becomes infeasible,

Algorithm 1: Trajectory Feasibility Check

```
if Status == 'Feasible' then
    X_last_success_input = X*_{j-1}
    X_last_success_output = X*_j
    X*_j = X_last_success_output
else
    X*_j = 0.5 * (X_last_success_input + X*_j)
end
```

we reduce the step size to maintain problem feasibility in order to handle the more nonlinear regions of the trajectory space.

2.4 Simulation

2.4.1 Apptronik$^{\text{TM}}$ Draco-Inspired System

The formulation in the previous section is applied to the two-link Draco robot (Fig. 2.2) in simulation. The Draco humanoid robot leg prototype is driven by viscoelastic actuators at its ankle and knee joints. Because the viscoelastic actuators used in the Draco system are very stiff, approximately $8e^6$ N/m, these elements offer minimal energy-storing capabilities. For this study, we explore the advantages of implementing softer springs in this system for a high-performance task.

The state space model in (2.20) is used with $p = 2$. The two actuators have equivalent spring, motor, and load dynamics. The Draco leg, excluding the actuation linkages, is essentially a two-link manipulator. The process to develop the dynamic equations of the robot to include the actuator states follows that described in Section 2.2. The variables F_1' and F_2' are obtained from Lagrangian dynamics with $M(q) \in R^{2\text{x}2}$, and $C(q, \ddot{q})$, $G(q) \in R^2$. Due to space limitations, the coefficients of $M(q)$, $C(q, \dot{q})$, and $G(q)$ for this two-link robot can be found in [20] and the code is available in my public repository [21].

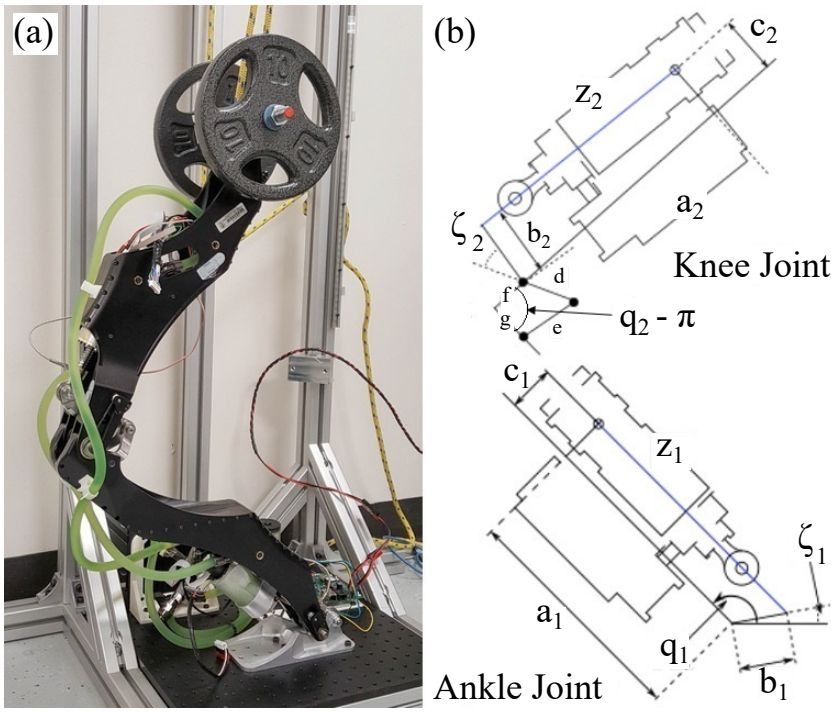

Figure 2.2: (a) Draco leg prototype. Our simulation-only experiments are modeled after this robot, with significantly reduced spring rates, performing the liftoff phase of a jump. (b) Schematics emphasize the nonlinear transmissions between actuator length, z, and joint angle, q, for the ankle and knee. These nonlinear transmissions motivate our choice to represent the robot impedance in actuator-length–actuator-force space rather than the standard joint-angle–joint-torque space.

To use (2.17), the moment arms, $L_1(q_1)$ and $L_2(q_2)$, of the ankle and knee joints, respectively, must be considered as:

$$L \triangleq \begin{bmatrix} L_1(q_1) & 0 \\ 0 & L_2(q_2) \end{bmatrix}. \tag{2.22}$$

We chose to demonstrate our algorithm for the goal of maximizing velocity at the center of mass (COM) of the robot, to obtain an optimal trajectory for a jumping motion. The parameters used for the simulation were guided by system identification of our lab's SEA and the parameters of the Draco leg. Select parameters are included

in Table 2.1. In the table, the parameters I_1 and I_2, m_1 and m_2, and l_1 and l_2 equal the moments of inertia, masses, and lengths of the lower and upper legs, respectively.

Table 2.1: Dynamics

M_S (kg)	1.7
k_S (N/m)	250k
β_S (Ns/m)	0
M_m (kg)	293
β_m (Ns/m)	1680
M_L (kg)	0
M_p (kg)	580
β_L (Ns/m)	0
I_1 (kg-m^2)	0.077
I_2 (kg-m^2)	0.050
$m_1(kg)$	3.77
m_2 (kg)	15
l_1 (m)	0.5
l_2 (m)	0.5

Table 2.2: Transmissions (Fig. 2.2.b)

a_1 (m)	0.21
b_1 (m)	0.04
c_1 (m)	0.02
ζ_1 (rad)	.464
a_2 (m)	0.2
b_2 (m)	0.05
c_2 (m)	0.04
d (m)	0.04
e (m)	0.03
f (m)	0.03
g (m)	0.01
ζ_2 (rad)	.524

Table 2.3: Constraints

$\overline{\delta}$ (m)	0.012
$z_{min,1}$ (m)	.1700
$z_{min,2}$ (m)	.1563
$z_{max,1}$ (m)	.2351
$z_{max,2}$ (m)	.2304
$\overline{\dot{y}}$ (m/s)	0.3
$\overline{u}$ (A)	15
$\overline{\Delta z}$ (m)	0.1
q_{1N} (rad)	1.96
q_{2N} (rad)	5.30
N	85
ΔT (s)	.0095

2.4.2 Ground Contacts

Ground contact wrenches are considered in the Draco model in the styles of [22][3], [23]. Point contacts with static Coulomb friction, with the coefficient of friction, $\mu = 0.8$, are applied: one at the front of the foot and one at the heel. Friction cones are formulated at each contact point using the basis vectors $b_1 = \begin{bmatrix}\mu & 1\end{bmatrix}^T$ and $b_2 = \begin{bmatrix}-\mu & 1\end{bmatrix}^T$. The positive force intensity parameters Φ_1, Φ_2, Φ_3, and Φ_4 are the basis vector multipliers, with two of these force intensities associated with each end of the foot, as shown in Fig. 2.3. Our linear program poses as equality constraints that the contact wrenches must satisfy Newton's second law in the x, y, and rotational directions. The force intensities must also be greater than or equal

[3]In our 2D simulations, this style of linear parameterization is not an approximation of the true friction cone, but it is in 3D space.

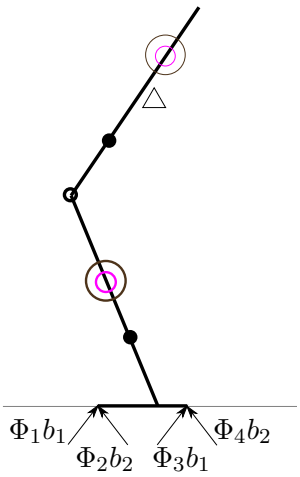

Figure 2.3: The simulated robot with point contacts at the front (left) and back (right) of the foot.

to zero until the robot jumps, as indicated in Section 2.3. These constraints imply a zero moment point condition [22].

2.4.3 Velocity Maximization for Jumping

In this study, the cost function to be minimized expresses the goal to maximize the upward y-velocity of the robot COM at the final time, $V^* \triangleq J_{\text{com\_y\_velocity}} \dot{z}_N$, where this Jacobian is known a priori due to our constrained final position, $z_{\text{fin.}}$. The simulation mimics the configuration shown in Fig. 2.2. We also strive to avoid unnecessary deviations from the motor current trajectory which keeps the robot at equilibrium with its springs, $\mathbf{U}_{baseline}$. (See Appendix A for more details.) We amend the cost function (to be minimized) to include the 1-norm of deviation from the baseline control signal, $\mathbf{U}_{dev} = \mathbf{U} - \mathbf{U}_{baseline}$. However, to keep the cost function linear, we create the variable matrix $\mathbf{U}_{abs}$ to represent $|\mathbf{U}_{dev}|$, as shown in (2.21). We have also added a slight preference towards solutions with small force intensities:

$$h(\mathbf{X},\mathbf{U}) = -J_{\text{com\_y\_velocity}}\dot{z}_N + \alpha \sum_{i\in\mathcal{P}} \sum_{n\in\mathcal{N}/N} u_{abs,i,n} + \\ + \gamma \sum_{i\in 4} \sum_{n\in\mathcal{N}/N} \Phi_{i,n}, \tag{2.23}$$

where α equals $1e^{-5}$ and γ equals $1e^{-8}$. This cost function is linear, supporting our problem structure. Considering (2.21), Φ is also an optimization variable in this problem.

For our simulation, the initial condition is at equilibrium with the two springs, which drives the formulation of $\mathbf{Z}_{base}$. (See Appendix A for more details.) The initial and final conditions capture that the leg position starts and ends at the same angular configurations, q_{1N} and q_{2N}. The final constraint is that the x-component of velocity at the COM is equal to zero at the final time.

The sequential linear optimization problem is solved using the Matlab CVX library [24] with the Gurobi solver. A time period of 0.798 s is considered. The algorithm converges in 19 iterations, $j = 19$, within a tolerance of 0.001 for $||\mathbf{X}^*_j - \mathbf{X}^*_{j-1}||_2$. Fig. 2.4 demonstrates exponential convergence of our iteration scheme. The corresponding behavior is shown in Fig. 2.5.a-2.5.c. An optimal value of 1.92 m/s upward velocity is achieved. One will notice spring oscillations, demonstrating the use of the two springs to store and release energy. Draco bends down and springs upward, following a jumping trajectory.

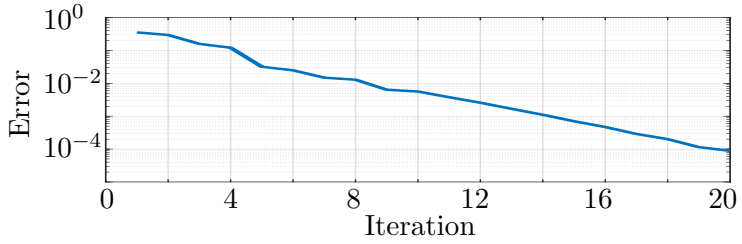

Figure 2.4: With the 25th iteration trajectory from the compliant jumping study as a baseline, the results suggest that the error decays exponentially.

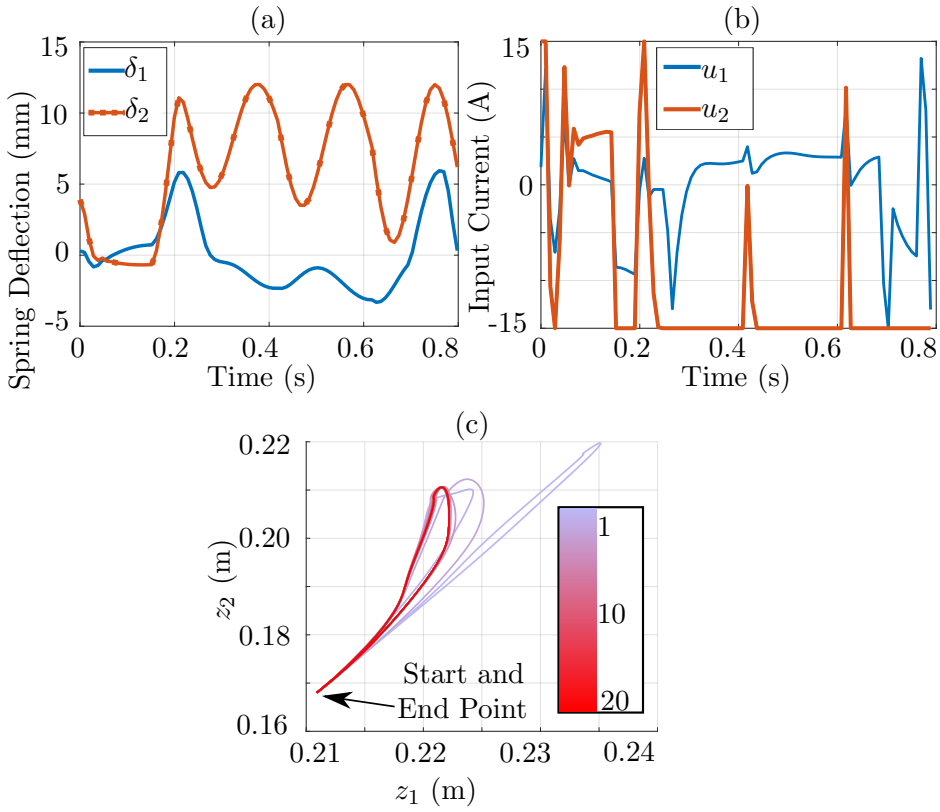

Figure 2.5: (a) Spring deflection trajectories for the compliant leg's optimal behavior. (b) The corresponding optimal u's to produce the optimal trajectory, which operate at the input limits. (c) The z trajectories produced over 20 iterations to produce the jumping behavior, demonstrating convergence.

This problem can also be formulated with the assumption of rigid actuators to allow for a direct comparison between the optimized trajectories for the rigid and compliant cases. Specifically, (2.6), (2.9), and (2.10) are used without considering the spring subsystem. The cost function in (2.23) is used for the rigid and compliant cases, and the resulting optimal motions are compared. Considering the same initial heights of the robots' COMs, the compliant leg's COM reaches a height that is 36% higher than that of its rigid counterpart. Fig. 2.6 shows the associated Matlab simulation with a comparison of the achieved COM heights. Fig. 2.7 shows that the

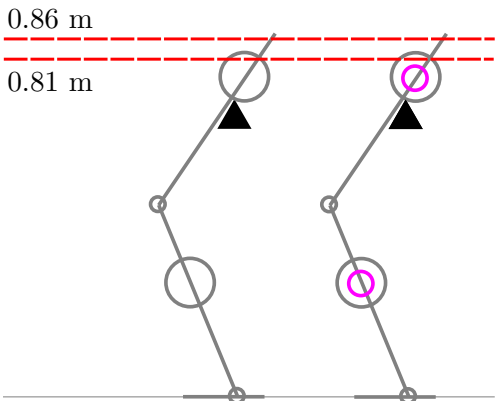

Figure 2.6: The rigid robot, left, and compliant robot, right (with the springs indicated in pink), after they jump and return to the ground. The COMs of the two robots are illustrated as black triangles. The two COM initial heights are both 0.67 m when the robots lift into the air, and the maximum heights of the compliant and rigid configuration COMs are 0.86 m and 0.81 m, respectively, which are marked with red lines.

optimal velocity in the compliant configuration, 1.92 m/s, is 16% greater than that of the the rigid configuration, 1.65 m/s. For the rigid robot, the ball screw limits are not reached, but its motion is still constrained by acceleration limits, damping, and ground contact constraints. These results demonstrate the gains that can be achieved from leveraging the dynamics of the springs.

Our optimization program for the rigid system converges in 25 iterations, as compared to 19 iterations in the compliant simulation. Table 2.4 shows the breakdown of average computation time per iteration to calculate F', A_{lin}, B_{lin}, *bias*, and the wrench components, and the time spent in the Gurobi optimizer. These results demonstrate that consideration of compliance introduces only slightly increased computational costs in our method.

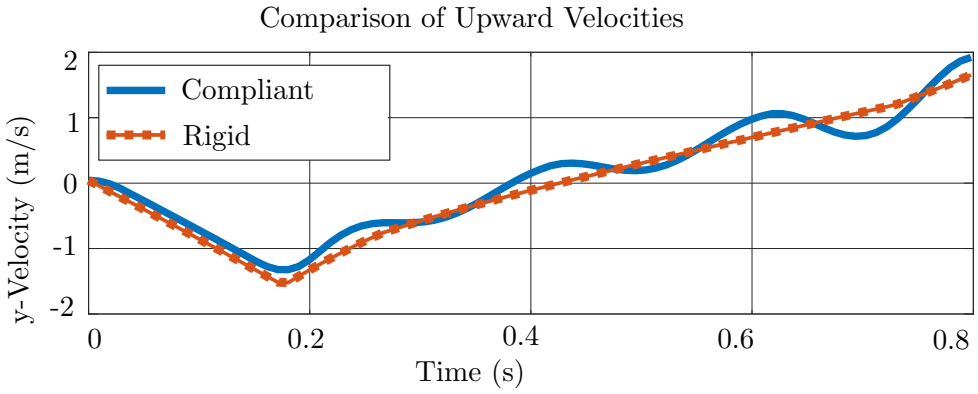

Figure 2.7: This comparison between the upward velocity components of the optimal trajectories for the rigid and compliant systems shows that the final velocity of the compliant system is 1.2 times that of the rigid system.

Table 2.4: Average Time per Iteration for Algorithm Components and Total Simulation Time (s)

Configuration	Linearization	Optimization	Total Time
Compliant	0.077	1.32	28.5
Rigid	0.072	1.14	32.1

2.4.4 Zero Input Behavior

To validate simulation accuracy, we ensured that energy was conserved throughout a zero input simulation. A test was conducted in which the system was released from rest from a nearly vertical position. The motors were off and no current was sent to the system. Compared to the jumping studies, the system was more heavily influenced by the changing transmission as the links fell downward due to gravity. As shown in Fig. 2.8, the algorithm converged quickly, in 12 iterations, even in this highly nonlinear case, thereby demonstrating its success in handling nonlinearities in the system.

To investigate energy conservation, a time period of 0.60 seconds is considered with $\Delta T = 1e^{-4}$ s to allow comparison to the numerically-difficult $M_p = 0$ case. A pseudo-mass value, $M_p = 580$ kg (reflected inertia from the robot space to the

actuator space through the nonlinear transmission), results in 0.55% total energy fluctuation during falling, whereas $M_p = 0$ results in 2.73%. With the time step used in the jumping trajectory generation, $\Delta T = 0.0095$ s, energy varies by 1.79% with the reasonable pseudo-mass. As M_p approaches 0, the robot does not even fall, which reflects a loss of important dynamics in the discretization process with this larger time step. At $M_p = 20$ kg and this same time step, energy fluctuates by 46%. These outcomes emphasize the importance of the M_p parameter in designing a reasonably discretized model.

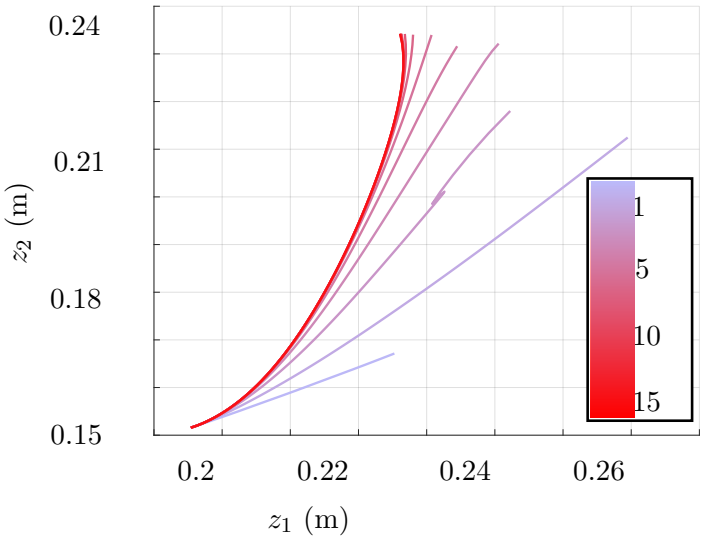

Figure 2.8: The z trajectories produced over 15 iterations ($M_p = 580$ kg and ΔT = .0095 s), showing convergence for the system's zero input behavior.

2.4.5 Pseudo-Mass Selection

The spring oscillation eigenvalue of the actuator system is influenced by the reflected link inertia, and can exceed the sampling rate (and therefore suffer from aliasing when discretized) in the absence of a tuned pseudo-mass parameter. M_p is set to 580 kg for the two actuators based on closeness to the largest eigenvalue in the expected operational range.

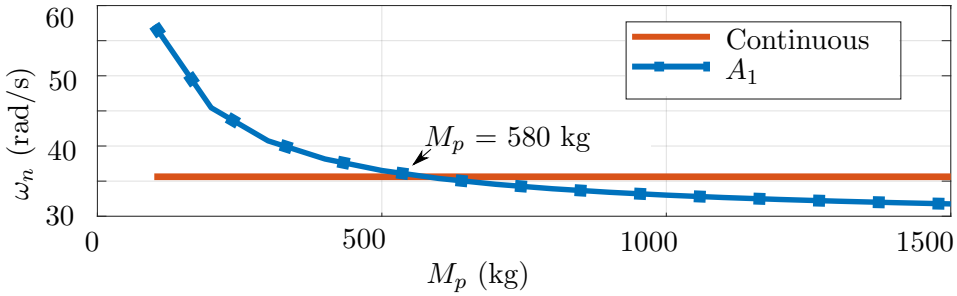

Figure 2.9: The natural frequency of A_1 (linear time-invariant actuator dynamics with pseudo-mass) varies with M_p, and we select an M_p value where the frequency aligns with that of the nonlinear, continuous dynamics at 35 rad/s.

Fig. 2.9 illustrates the importance of selecting a reasonably-tuned M_p value. The 'Continuous' eigenvalues, which are independent of M_p, represent the full dynamics of the nonlinear system. With a tuned value of M_p, the full dynamics approximation used for optimization (A_{lin}) will align closely with the actual dynamics. The figure demonstrates that the penalty for choosing an M_p value too small, or neglecting it entirely, is greater than for picking a value that is larger than 580 kg. This is because, if M_p were equal to zero, the actuator model's spring dynamics would alias when discretized. If M_p approached infinity, this would equate to a model with infinite output impedance, which introduces error, but is a common modeling assumption for SEAs. Since the time step, ΔT, and associated sampling frequency used to discretize the system in Section 2.2.1 must be significantly greater than the largest eigenvalue of the continuous system to avoid aliasing, the pseudo-mass modification is essential to allowing large time-steps, small linear program sizes, and fast run-times.

2.5 Algorithm Tuning

We also experimented with providing our algorithm with a better-informed nominal trajectory. Specifically, we used the optimal trajectory of the rigid system as the nominal trajectory for the compliant leg. With the same tolerance (0.001), the algorithm converges in 17 iterations rather than 19. This result supports only a small efficiency gain from using rigid system trajectory optimization as a warm start heuristic for actuator-centric trajectory optimization. We expect that in practice, actuator-less kinematic trajectory planning will be a critical first step even if actuator-less trajectory optimization is not a useful warm-start.

Fig. 2.10 shows the results of modifying the input penalization parameter, α, as defined in (2.23). The results show the expected decrease in optimal velocity after reaching a threshold value, as well as success in algorithm convergence for various tuned cost functions.

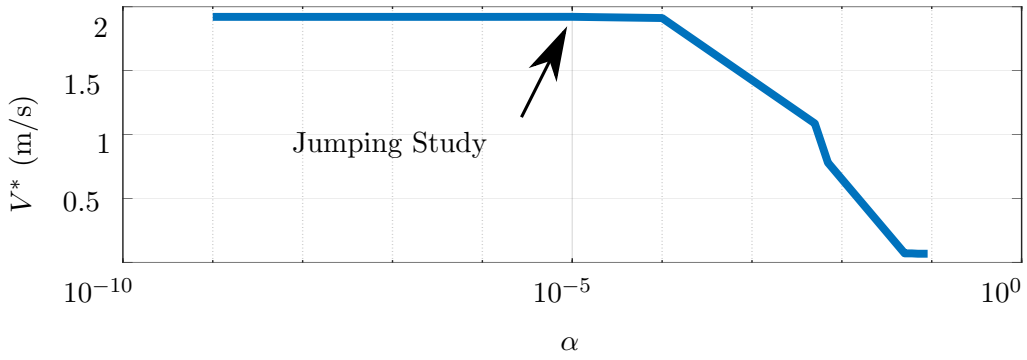

Figure 2.10: Our algorithm converges as α is tuned, demonstrating the limits on optimal upward COM velocity, V^*, as the penalty increases.

The performance of our approach depends strongly on the subproblem solver. While CVX in Matlab offers several solvers, we found that using a solver specifically suited for linear programming, Gurobi, was needed to extend our approach to larger time scales.

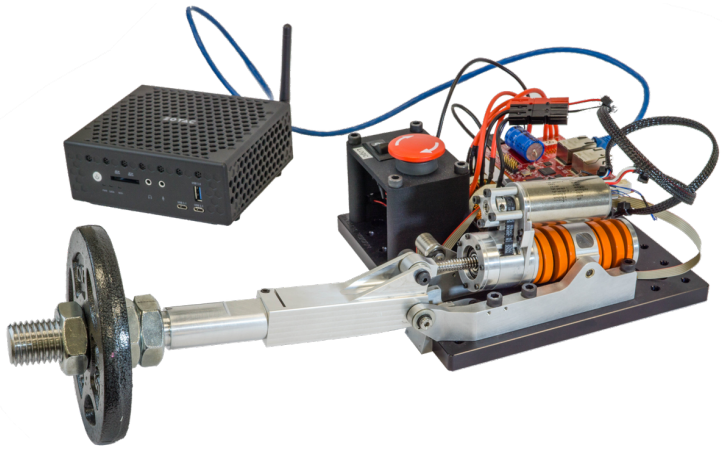

Figure 2.11: The Taurus testbed, with an SEA whose spring is softer than Draco's viscoelastic elements by an order of magnitude. For our experiment, a 5 lb weight is attached to the actuator arm.

2.6 Experiments

The findings from simulation were applied for validation on the single degree-of-freedom Apptronik Taurus testbed with the P170 Orion SEA (Fig. 2.11). The state space model in (2.20) was used with $p = 1$. Our trajectory optimization scheme relies on a well-identified model, so that the control system can depend heavily on the feed-forward, open loop command for high-speed tasks. System identification was performed using a least squares approach by fitting the parameters in (2.6) to the system's response to white noise and chirp signal current inputs. The identified parameter values are outlined in Table 2.5.

The goal of the experiments was to maximize final joint velocity at 0.52 seconds. This corresponds to a negative final actuator velocity, since $L < 0$ for this system. The cost function to be minimized is:

$$h(\mathbf{X}, \mathbf{U}) = \quad \dot{z}_N + \sigma \sum_{n \in \mathcal{N}/N} u_{abs,i,n}, \tag{2.24}$$

where σ equals $1e^{-8}$. In addition to our feedforward current command, we implemented a simple P controller, feeding back motor position. For increased stability we

Table 2.5: P170 Identified Parameters

M_S (kg)	1
k_S (N/m)	698600
β_S (Ns/m)	500
M_m (kg)	250
β_m (Ns/m)	5885
M_L (kg)	0.227
M_p (kg)	220
β_L (Ns/m)	0

Table 2.6: P170 Constraints

$\overline{\delta}$ (m)	0.01
z_{min} (m)	.0911
z_{max} (m)	.1389
$\overline{\dot{y}}$ (m/s)	0.3
$\overline{u}$ (A)	3
$\overline{\Delta z}$ (m)	0.1
q_N (rad)	1.57
z_N (m)	0.11597
N	105
ΔT (s)	.005

controlled motor position, rather than joint position, in order to control a collocated system from the control input [25]. Because our configuration involves feedback, it is important for safety to ensure that conservative current limits are used in the optimization scheme, and that the software used for implementation upholds the hardware's actual upper limits. In this experiment an optimal trajectory was produced with a maximum allowable current of 3 A, but the motor saturation limit was 8 A. Constraints for optimization are shown in Table 2.6. A smaller time step was used in this optimization scheme to support the convergence of the specific problem.

For one experiment, the optimal trajectory is devised with spring dynamics considered, and for comparison, an optimal trajectory is produced while ignoring spring states. Trajectory generation is first performed in simulation using CVXPY [26] (to explore available solvers), to obtain the feedforward current command and desired trajectory. The difference in computational costs for the compliant and rigid systems are negligible in this case: four iterations and six seconds with compliance considered versus five iterations and six seconds without compliance.

To select a reasonable pseudo-mass[4], we plotted the maximum eigenvalues of the continuous system, and the maximum eigenvalue of A_1 for several distinct M_p values. We chose $M_p = 220$ kg based on the alignment of the largest eigenvalue

[4]Approximating reflected inertia of the arm w.r.t. actuator displacement.

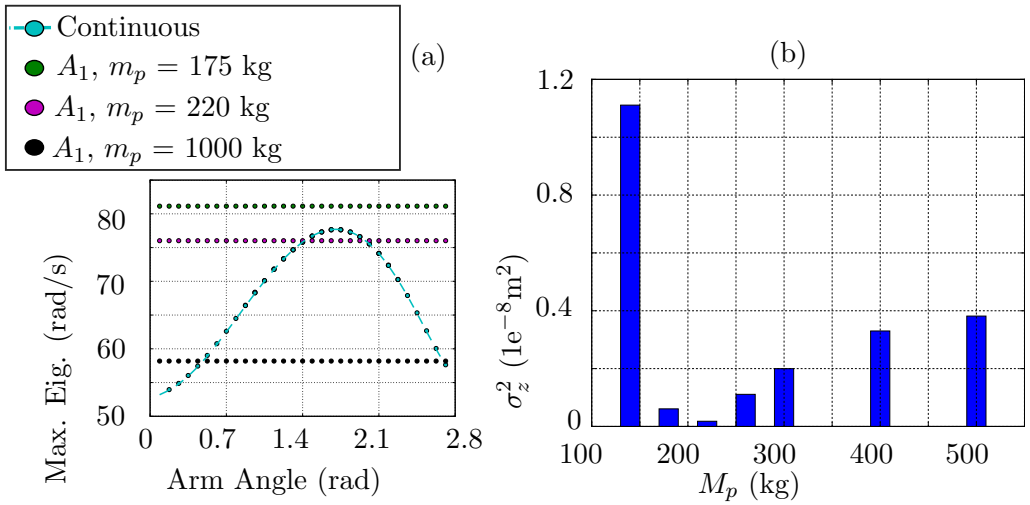

Figure 2.12: Tuning pseudo-mass. (a) Maximum eigenvalue frequency for true arm-actuator system (Continuous) and various approximations (A_1) for the actuator alone with varying pseudo-mass (M_p). The arm points down at 0 rad. Changing eigenvalue frequency for the true system is due to angle-dependent reflected inertia. The choice of 220 kg is relatively accurate in the operational region centered around 1.57 rad. (b) Simulation error of discrete-time models used for trajectory optimization as a function of pseudo-mass, for a feedforward trajectory in the operational region, with a fixed time step. Average squared error relative to the trajectory of the true model. The low pseudo-mass example has aliasing errors. High pseudo-mass errors exist, but are not as extreme.

over a range of likely arm configurations, as seen in Fig. 2.12.a. To quantify the error between the continuous dynamics and the approximated dynamics with a particular pseudo-mass, we obtained the $\mathbf{Z}$ trajectories for the continuous, true dynamics, $\mathbf{Z}_c$, and the approximate, linearized dynamics, $\mathbf{Z}_{lin}$, for a pre-defined input current trajectory in the expected operating region. The error associated with our pseudo-mass selection can then be expressed by the mean squared error of the $\mathbf{Z}_{lin}$ trajectory, σ_z^2, as seen in Fig. 2.12.b.

Fig. 2.13 shows the results of the experiments. To compare the expected and actual behaviors, the position states of the simulated and experimental data sets are filtered using a second order Butterworth filter with a cutoff frequency of 30 Hz.

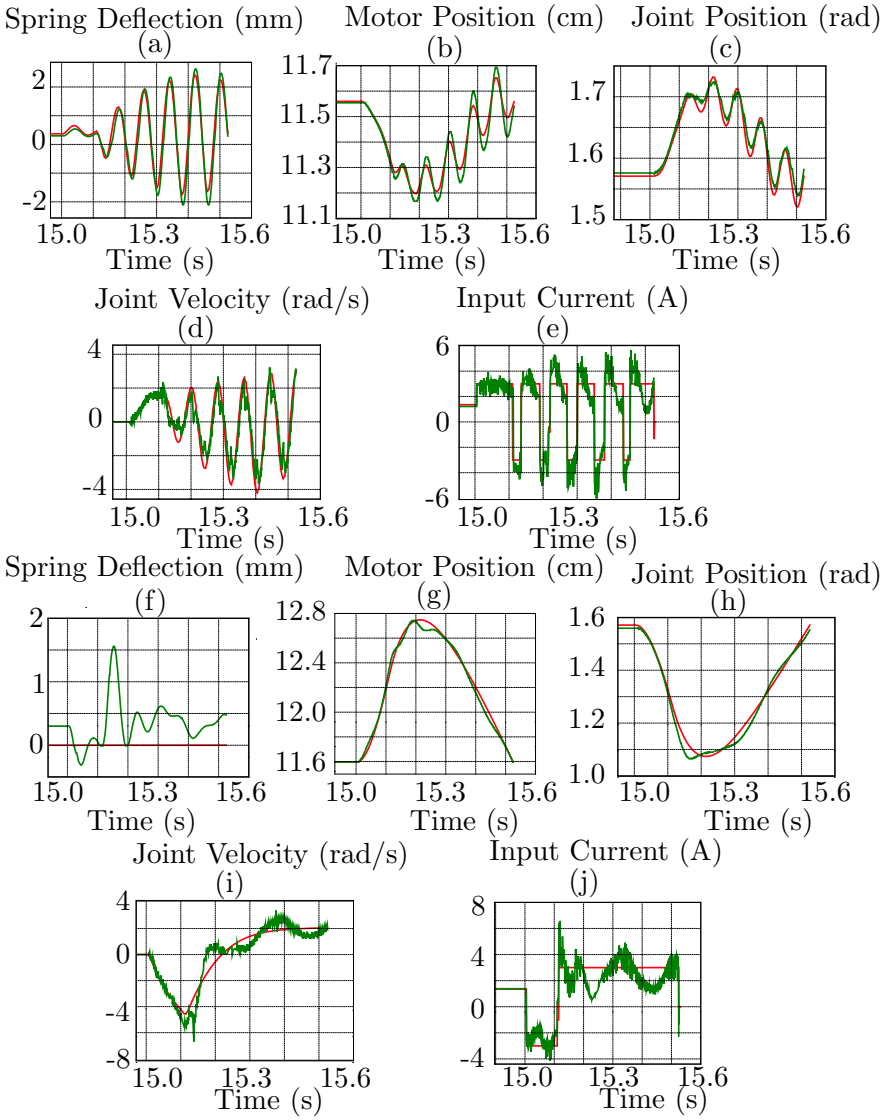

Figure 2.13: (a)-(e) show the simulated (red) and actual (green) optimal behavior of the P170 actuator when spring dynamics are considered. The high-quality tracking in (a)-(e) support that the system has been well identified. (f)-(j) show the expected and actual behaviors of the actuator when spring dynamics are not considered. These results demonstrate dynamic inconsistencies in tracking when the spring subsystem is neglected in planning. The experimental data are shifted by 5 ms to account for time delay. Fifteen seconds are used to interpolate to the starting position, and the optimal motion begins at 15 s.

Table 2.7 summarizes the actual and ideal results. In both experiments, the actuator is able to start and end at the desired actuator position, .11597 m, with negligible error. With compliance modeled, the optimal trajectory is oscillatory in order to store and release energy, while the optimal trajectory for the rigid counterpart is a down-up motion. While there is 0.06% error in the final velocity when compliance is considered, there is 8.21% error in the final velocity when the system is considered rigid. Fig. 2.13 (e) and (j) show that the feedforward current aligns well with the actual required current when spring dynamics are considered, while there are more deviations from nominal when the system is considered rigid. When compliance is modeled, the ideal, final optimal velocity is 51% greater than that achieved with the rigid model. These results demonstrate the benefit of modeling compliance for dynamically feasible motions, and the gains of leveraging compliance for high-performance tasks.

Table 2.7: Experiment Results

Configuration	Actual z_N (m)	Ideal $\dot{z}_N$ (m/s)	Actual $\dot{z}_N$ (m/s)
Compliant	0.11570	0.07671	0.07666
Rigid	0.11644	0.05094	0.04676

2.7 Discussion

Actuator dynamics are often neglected from robot motion planning due to computational complexity, and our proposed method for trajectory optimization offers several advantages in this regard. First, directly capturing all relevant state and input constraints is an essential feature for a dynamically consistent trajectory. Our new robot–actuator interface, modified by pseudo-mass M_p, allows us to exploit the structural difference between a linear actuator admittance and a nonlinear robot impedance—which is novel and efficient. Through our formulation, we can increase the states of the system to include actuator dynamics without paying the

computational cost typically associated with adding states to nonlinear optimization problems. Finally, we have demonstrated the gains in executing a high-performance task by leveraging compliance in the linear optimization subproblem. In the future, we are interested in how our actuator model could add on to any standard robot impedance model, including the very general constrained floating base model.

Chapter 3

Toward Achieving Formal Guarantees for Human-Aware Controllers in Human-Robot Interactions

3.1 Introduction

As robots become more embedded into our everyday lives and begin to collaborate with humans, a large potential emerges to boost human productivity by eliminating unnecessary human chores in workplaces [27]. This potential can only be realized by robot control systems that process and react to human needs.[1] A survey of human-aware robot navigation shows that many researchers have studied motion and task plan generation for socially-aware robots, for activities ranging

[1]This chapter contains material from [28]. My main contributions include model and software development for controller synthesis and high-level controller implementation for the hardware experiments.

from operating in human-occupied areas to engaging in social cues [29]. However, these methods often lack robustness to disturbances and/or formal guarantees of goal achievement. Formal methods have been leveraged in robotic applications, but there is a growing need to apply these techniques to robots that continuously and directly interact with humans. Our work takes a step toward addressing this need.

Reactive synthesis has been applied in contexts where disturbances are unavoidable, such as the DARPA Robotics Challenge [30] and other challenging setups [31], to provide formal guarantees for specification realizability. There is a gap in robotics and formal method literature with respect to applying these methods to direct human-robot interaction (HRI). In reactive synthesis problems, humans are often framed as randomly or periodically interfering with the robot's goal [31], [32]. In [33], a reactive synthesis problem is formulated to generate a policy for a robot to reach a goal position in a simulated kitchen scenario, but the only human interaction involves avoiding two moving chefs. We seek to be robust to a larger variety of disturbances, and to generate policies in which humans and robots continuously interact.

There is much interest in the HRI community for robot controllers to consider human factors to boost human productivity [34], [35]. Several research groups are exploring formal verification methods for robots to interactively support humans [36], but few groups have incorporated human factors into these methods. In [37], the authors verify whether a robot assistant can reach commanded positions and deliver medicine. In addition to these types of interactions, we also explore additional knowledge of human requirements to improve collaborative task execution. Ref. [38] takes a step in this direction: A cognitive model of trust is incorporated into a stochastic multi-player game and probabilistic rational temporal logic specifications are proposed, but probabilistic model checking is left as future work. In [39], social norms for a hand-off task are represented as transition systems, and model checking

is performed to verify successful task completion. In contrast to this work, our framework uses reactive synthesis to prevent specification violations, while being robust to uncertainties.

Although it is impossible to generalize human behavior, works like [40], [41] still demonstrate the insight to be gained by considering human models for decision-making. For example, in [42], hypothetical human models that consider human proficiency and stress are employed to synthesize paths for semi-autonomous operation of drones with human operators. Similarly, we focus on studying the implications of incorporating human models. Our approach provides the flexibility to update the human model for effective and personalized human-robot interactions.

The goal of this chapter is to demonstrate a proof-of-concept for devising control policies via reactive synthesis that consider and improve human working behavior. In doing so, we generate a controller that is robust to disturbances and provides formal guarantees for specification satisfaction in an HRI scenario. The main contribution of our work is a study on reactive synthesis that incorporates human factors for human-aware robot cooperative tasks. We consider an HRI case study in which we construct a model of human workers in a workplace as transition systems to devise a robot controller to deliver and pick up work while considering the human's needs. To this end, we formalize system specifications using linear temporal logic. We use reactive synthesis to automatically construct a controller that meets all system specifications and a human's productivity needs. We then demonstrate the reactive controller on the Toyota HSR (Fig. 3.1). Ultimately, we explore the question of how robots can make humans more productive by limiting unnecessary human tasks.

3.2 Preliminaries

Our notation employs the formalisms of [43], which are summarized below:

Figure 3.1: (a) The Toyota Human Support Robot, which we use as our experimental platform for human-informed work delivery, picking up completed work. (b) The HSR dropping off completed work at the Inventory Station.

Definition 1: A transition system, TS, is a tuple $TS = (S, Act, \rightarrow, I, AP, L)$ where S is a set of states, Act is a set of actions, $\rightarrow \subseteq S \times Act \times S$ is a transition relation, $I \subseteq S$ is a set of initial states, AP is a set of (Boolean) atomic propositions, and $L : S \mapsto 2^{AP}$ is a labeling function.

Definition 2: An infinite path fragment, $\pi = s_0 s_1 s_2 ...$, for $s_i \in S$, is an infinite sequence of states such that $s_{i+1} \in \{s'_i \in S : \exists \alpha \in Act \mid s_i \xrightarrow{\alpha} s'_i\} \, \forall i \geq 0$. An infinite path fragment is a path if the initial state, $s_0 \in I$. The set of paths in TS is denoted as $Paths(TS)$.

Definition 3: The trace of π, $trace(\pi) = L(s_0)L(s_1)L(s_2)...$, is a sequence of sets of atomic propositions that are true in the states along the path. The set of traces of TS is defined by $Traces(TS) = \{trace(\pi) : \pi \in Paths(TS)\}$

Definition 4: A linear-time (LT) property, P, over atomic propositions in AP, is a set of infinite sequences over 2^{AP}. TS satisfies P, represented by $TS \models P$, iff $Traces(TS) \subseteq P$.

Definition 5: Linear-temporal logic (LTL) is a formal language to represent LT properties. The operators used in this chapter to construct LT formulas

are conjunction ($\wedge$), disjunction ($\vee$), next ($\bigcirc$), eventually ($\Diamond$), globally ($\Box$), implication ($\rightarrow$), and negation ($\neg$). Let Φ be an LTL formula over AP. TS satisfies Φ, represented by $TS \models \Phi$, iff $\pi \models \Phi$ for all $\pi \in Paths(TS)$.

3.3 Work Delivery with Human Backlog Model

This case study examines a robot operating in a work environment and is inspired by [44]. The robot drops off new work ("deliverables") at the Human Workstation, picks up completed work from the Human Workstation, and drops the completed work off at the Inventory Station. The robot's contributions thereby eliminate unnecessary movement by the human to pick up and drop off work. The goal is for the robot to operate with an awareness of the human's backlog, defined as the amount of uncompleted work at the Human Workstation. Assessing human backlog is an ongoing area of research [45]. We take backlog to be an indicator of stress levels, as too little work can cause boredom and too much work can result in higher levels of frustration [46]. We seek to synthesize a controller that guarantees, despite system disturbances, that the human always has work to complete and is not over-stressed by work demands.

3.3.1 Modeling

Human Model

In this scenario, the human is always present in the Human Workstation. (Work breaks are addressed in Sec. 3.4.) The human's backlog, BL, can range from 0% to 100%, relative to the maximum amount of uncompleted work that can be present in the workstation. When the robot is not present in the Human Workstation, the human works whenenever there is uncompleted work. We consider a simple, discrete linear BL model that is a function of ΔT time steps that each last t_d seconds:

$$BL(\Delta T) = BL_{init} - \gamma \Delta T,\ \Delta T \in \{0, 1, 2, ...\}, \tag{3.1}$$

where BL_{init} is the initial amount of backlog at the Human Workstation, and γ is the work reduction rate per t_d seconds. The value of BL_{init} is updated each time the robot comes to the Human Workstation to deliver work.

We now formalize the way BL may change between time steps. There is uncertainty in how much BL decreases during each state transition, as the reduction value depends on how much time the robot requires to transition between states in the real-world execution. The worker's BL can decrease by integer multiples of γ each time step. We assume that BL may decrease by up to 5γ, based on the maximum amount of time the robot requires for its most challenging manipulation task. We consider two possibilities for how BL may shrink. When the robot is traveling between locations in the workspace, we define the formula v_1 such that the following holds:

$$\bigcirc v_1 \triangleq \bigvee_{k=0}^{2} (BL - k\gamma). \tag{3.2}$$

It may require more time for the robot to drop off completed work at the Inventory Station than to travel between locations, and so we allow for greater BL reductions between time steps for this task:

$$\bigcirc v_2 \triangleq \bigvee_{k=0}^{5} (BL - k\gamma). \tag{3.3}$$

The human transitions from the "work" state to the "wait" state if the human has completed all of her work and the robot has not yet arrived to deliver more work. The human transitions to the "refill" state when the robot is present in the Human Workstation and delivers more work. The robot being in the Human Workstation is equivalent to the robot state, RS, being equal to N. When the robot arrives at the Human Workstation, BL grows by δ. The human then returns to the working state

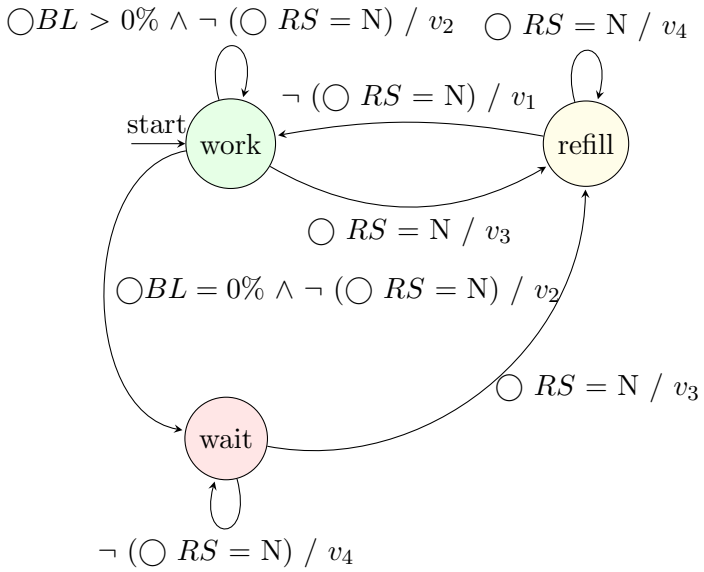

Figure 3.2: Human Model with three states. Transitions are triggered by guards, g, and there are corresponding outputs, y. In this transition system, edges are labeled in a g / y format. In this system, the outputs are described by v_2, v_3, and v_4.

when the robot departs from the workstation. As seen in Fig. 3.2, there are guards based on BL and robot behaviors which determine allowable state transitions. The BL variable is tracked, and grows and shrinks according to v_1, v_2, v_3, and v_4. The formula v_3 captures BL growth when the robot arrives at the Human Workstation:

$$\bigcirc v_3 \triangleq BL + \delta, \tag{3.4}$$

and v_4 captures when BL does not change between time steps:

$$\bigcirc v_4 \triangleq BL. \tag{3.5}$$

Fig. 3.2 demonstrates the possible non-determinism in how BL changes with each time step, and planning for this uncertainty is discussed further in Sec. 3.3.2.

Robot Model

The robot moves in a 1D grid. There are N+1 grid spaces, with the grid spaces labeled as 0 through N. Space 0 corresponds to the Inventory Station, and (3.3) is valid in this location when the robot drops off completed work. As discussed in the previous section, space N is the Human Workstation. The robot's actions are Go_{Sj}, which indicate that the robot is currently moving to position j $\in$ 0,1,...,N in the grid. The robot is free to move within this grid, except for when there is an obstacle present in a grid space blocking a path. We define the atomic proposition, "an obstacle is present in State j," as $\mathcal{O}_j \ \forall j$ = 1,2,...,N-1. The transition system is shown in Fig. 3.3.

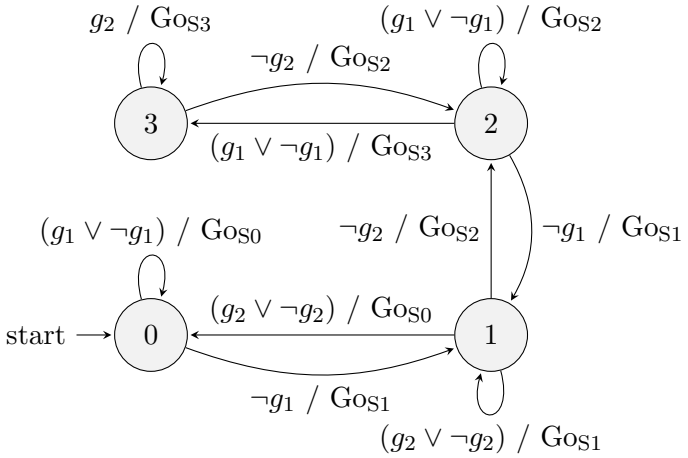

Figure 3.3: Robot Model with N = 3. State 0 is the Inventory Station and State 3 is the Human Workstation. Each edge of the TS is labeled with a guard and an action. The presence of an obstruction in one of the robot's adjacent positions can restrict the robot's next action. The guards express whether or not there is an obstacle blocking the robot from proceeding to a neighboring state, and we define $g_1 \triangleq \mathcal{O}_1$ and $g_2 \triangleq \mathcal{O}_2$

Full HRI System Model

A transition system is then formulated to capture that the robot drops off deliverables and picks up completed work at the Human Workstation, and drops off completed work at the Inventory Station. The robot also operates with an awareness of the human's work backlog, which will allow for controller synthesis that considers BL. To combine and synchronize the human and robot systems, the two separate human and robot models were used to create states which represent both the human and robot at each time step. The transition system, shown in Fig. 3.4, is expressed as $TS_{HRI} = (S_1, Act_1, \rightarrow, I_1, AP_1, L_1)$ where:

- $S_1 = \{j_{work}, j_{wait}, N_{refill}\}\ \forall\ j$ = 0,1,...,N-1
- $Act_1 = \{\ \mathrm{Go_{Sj}}\ \}\ \forall\ j$ = 0,1,...,N
- $I_1 = \{0_{work}\}$
- $L_1(0_{work}) = L_1(0_{wait})$ = Robot is at Inventory Station.
- $L_1(N_{refill})$ = Robot is at Human Workstation.

3.3.2 Reactive Synthesis

It is necessary to incorporate robustness to uncertainty in our approach in order for the robot to pick up completed work, drop off deliverables, and reason about the human's BL in a real environment. We consider a two-player game in which the robot's actions are controllable. Obstacle interference, success of dropping off completed work, and the backlog reduction rate act as the uncontrollable environment. The robot and the environment take turns executing actions, and we seek to automatically synthesize a robot controller strategy that allows a system specification to be realizable despite any antagonistic actions executed by the environment. To meet the system requirements while handling external disturbances,

we formulate this scenario as a reactive synthesis problem in which plant actions are controllable and environment actions are uncontrollable. We seek to find a strategy that will uphold a specification no matter how the environment selects its actions for all time [47].

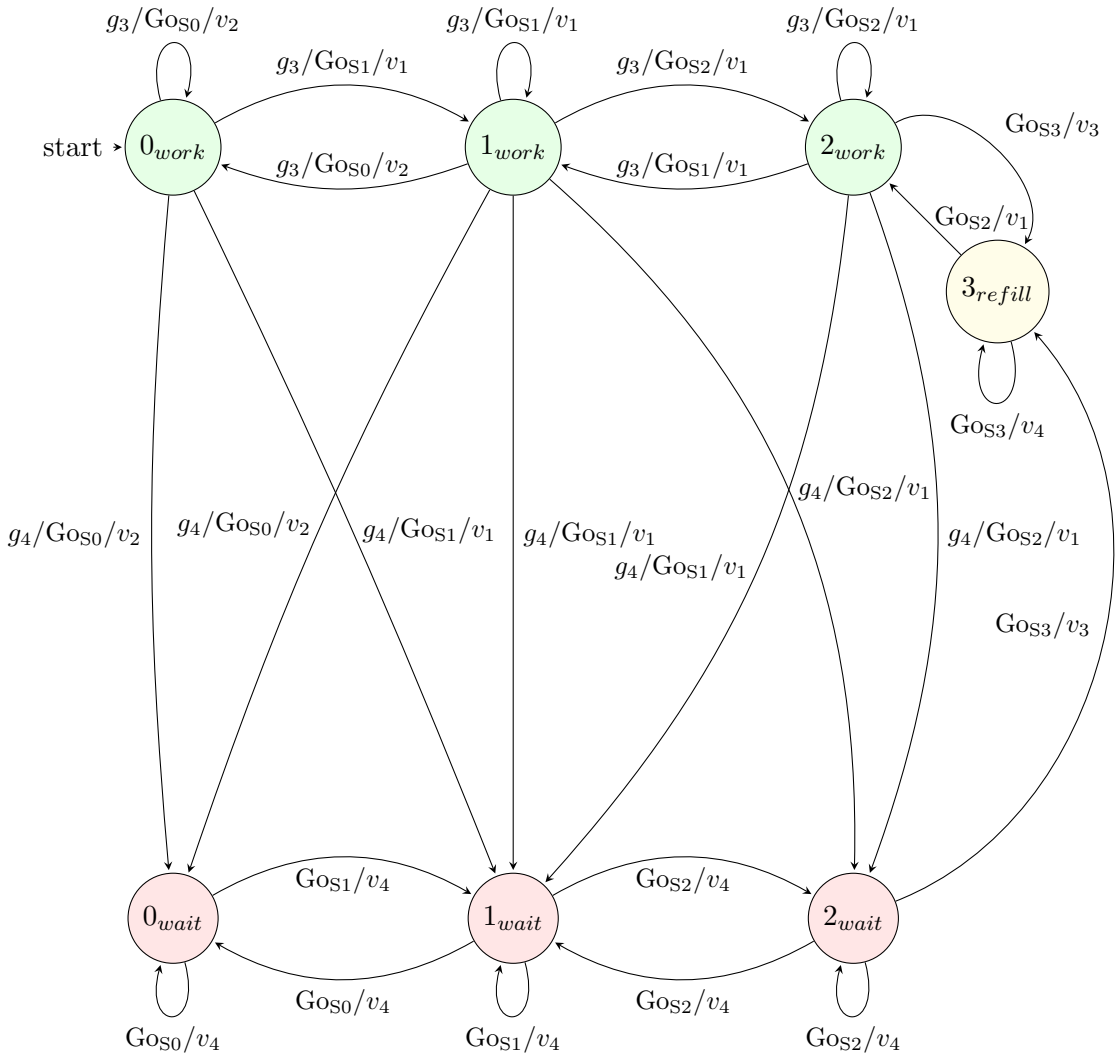

Figure 3.4: HRI Work Delivery and Pickup Transition System with N = 3. The edges of the transition system are labeled first by guards (g_3, g_4), then by actions, and lastly by pertinent output variables (v_1, v_2, v_3, v_4). The guards which determine whether the human is working or waiting are expressed by $g_3 \triangleq \bigcirc BL > 0\%$ and $g_4 \triangleq \bigcirc BL = 0\%$. As illustrated in Fig. 3.3, the robot cannot move to a neighboring state if it contains an obstacle, but we do not show this guard due to space limitations.

To automatically synthesize a controller, we must formalize the specifications that describe the possible environment behaviors. It is possible in a work environ-

ment that people may pass through states $\mathrm{S}_d = \{j_{work},\ j_{wait}\}\ \forall j : 0 < j < N$ and obstruct the robot from proceeding into an adjacent position in the workspace. We impose as a safety specification on the robot that it will not proceed toward a position that contains an obstacle. We also assume that if the human sees the robot moving to a workspace position, she will not intentionally move to block the robot's desired workspace position:

$$\Phi_{d1} \triangleq \bigcirc RS = j \rightarrow \bigcirc \neg \mathcal{O}_j \quad \forall j = 1,2,..\text{N-}1. \tag{3.6}$$

In our implementation, if there is a human occupying one of the robot's neighboring positions, the robot will ask her not to block the workspace. Thus, we assume that the person will move out of the way by the next time step:

$$\Phi_{d2} \triangleq \mathcal{O}_j \rightarrow \bigcirc \neg \mathcal{O}_j \quad \forall j = 1,2,..\text{N-}1. \tag{3.7}$$

We also account for the possibility that the robot may not successfully drop off completed work in the Inventory Station on its first try. At all robot states, the robot either is or is not manipulating completed work ("hand full" $\triangleq HF$ is true or false). When $RS = 0$, if the robot is currently manipulating completed work, it will be successful ($\mathcal{S}$) or unsuccessful ($\neg\mathcal{S}$) at dropping off the completed work in that time step. In order to prevent the environment from interfering indefinitely with a successful dropoff, we assume that no more than two consecutive tries are required to be successful. The associated specifications are written as follows:

$$\Phi_{d3} \triangleq (\neg(RS = 0) \wedge \bigcirc RS = 0 \wedge HF) \rightarrow \bigcirc(HF \wedge \text{tries} = 1 \wedge (\neg\mathcal{S} \vee \mathcal{S})), \tag{3.8}$$

$$\Phi_{d4} \triangleq (RS = 0 \wedge HF \wedge \neg\mathcal{S} \wedge \text{ tries} = 1) \rightarrow \bigcirc(HF \wedge \text{tries} = 2 \wedge \mathcal{S}), \text{ and} \tag{3.9}$$

$$\Phi_{d5} \triangleq (RS = 0 \wedge HF \wedge \mathcal{S}) \rightarrow \bigcirc(\neg HF). \tag{3.10}$$

As discussed in Sec. 3.3.1, when the robot is not dropping off deliverables,

there is uncertainty in how much BL will decrease as the robot transition between states in the real environment. This uncertainty will impact the value of BL at the next time step when the human is working. To express the specifications for BL reduction, we define a formula that describes the situations in which the robot will attempt to drop off completed work:

$$g_5 \triangleq \left[\{\neg(RS = 0) \wedge \bigcirc RS = 0\} \vee (\text{tries} = 1 \wedge \neg\mathcal{S})\right] \wedge HF. \tag{3.11}$$

We now distinguish between the robot moving within the workspace and performing the dropoff behavior. For workspace motions, we define

$$\Phi_{d6} \triangleq \neg(\bigcirc RS = \text{N}) \wedge \neg g_5 \wedge \neg\text{wait} \rightarrow \bigcirc v_1, \tag{3.12}$$

where v_1 is as defined in (3.2). During dropoff at the Inventory Station the following specification holds:

$$\Phi_{d7} \triangleq g_5 \wedge \neg\text{wait} \rightarrow \bigcirc v_2, \tag{3.13}$$

where v_2 is as defined in (3.3). We desire that the robot always eventually drops off completed work at the Inventory Station. Unless BL decreases, there would be no guarantee that the robot would always eventually have completed work to pick up from the human and drop off at the Inverntory Station. We add the assumption that if BL stays constant in two consecutive time steps, then BL will decrease in the next time step:

$$\Phi_{d8} \triangleq \neg(\bigcirc RS = \text{N}) \wedge v_4 \wedge \neg\text{wait} \rightarrow \bigcirc \left(\bigvee_{k=1}^{5} (BL - k\gamma)\right), \tag{3.14}$$

where v_4 is as defined in (3.5). We synchronize the real robot's motion with the BL model so that this assumption is valid in our implementation.

3.3.3 Controller Synthesis

The reactive synthesis problem was implemented using Slugs [48] with N = 3. (We consider four states for our proof-of-concept hardware implementation in Sec. 3.4, but the underlying techniques of our approach can handle a much larger number of states.) By formulating the problem as a two-player game, Slugs can construct a reactive robot controller that upholds our specification of interest within TS_{HRI}.

In the simulation, the robot starts at State 0, $\gamma = 3.3\%$, and $\delta = 50\%$. In order to strike a balance between state space fineness and computational efficiency, BL is represented in Slugs as 0,1,2,...,30, which corresponds to 0%,3.3%,6.7%,...,100%. Slugs was used to synthesize a controller that always satisfies the specification that the human's backlog never reaches 0% and never exceeds 87%. In this manner, the human always has work to complete, but the robot does not seek to stress her. It is also desired that the robot will return to the Inventory Station infinitely often to drop off completed work. Since BL can vary from 0 to 30 in Slugs, we express the system specification as:

$$\Phi_1 = (\square BL \leq 26) \wedge (\square BL > 0) \wedge \square\lozenge(RS = 0 \wedge HF) \tag{3.15}$$

We provide as an initial condition that BL_{init} is between 30% and 86.7% ($9 \leq BL_{init} \leq 26$), as we found that outside of this BL_{init} range, (3.15) is not realizable. We synthesize a strategy using a quad-core Intel Core i7 processor and 12GB of RAM. Slugs computes in less than five seconds that the specification is realizable, and devises a high-level controller that guarantees that the robot will react properly to its environment while upholding the system specification. The controller is in the form of a decision tree, with nodes that capture all possible combinations of robot and environment behaviors, and the possible transitions from each node. Based on the robot's present state and the environment's behavior, the decision tree determines the appropriate next robot action. We now have a policy

that can be leveraged for online decision-making.

3.4 Experiments

3.4.1 Toyota Human Support Robot

The Toyota HSR, which comprises an omni-directional base and a 5-DOF single manipulator, was adopted as the hardware platform for experimentation. The HSR uses two different computers: The main PC is for primary perception, navigation, and manipulation tasks. An Alienware laptop (Intel Core i7-7820HK, GTX 1080) is used for running OpenPose[2], a real-time convolution neural network based algorithm used for human detection. All robot sub-programs communicate with each other via the Robot Operating System (ROS) interface. An overview of the HSR skills used for controller implementation is provided below.

Perception

A laser range scanner is used on both sides of the mobile base to detect whether there is an obstacle within approximately one meter of the base. To simulate a more realistic working environment, the robot also perceives if there is a human in the workstation or if she has left to take a break. A depth camera for RGB-D video streaming is located on the HSR's head. Recognition of whether or not there is a human in the workstation, based on the RBG-D data, is executed by OpenPose. We created a ROS action so that the HSR turns its head toward the workstation every five seconds to check for worker presence.

[2]https://github.com/CMU-Perceptual-Computing-Lab/openpose

Navigation

All basic navigation functions, including wheel-joint control and avoiding obstacles, are included in the ROS navigation stack. It is assumed that given a goal position, the robot can safely navigate to this location, while avoiding dynamic obstacles via a re-planning scheme.

Manipulation

The robot's manipulator is used to pick up completed work from the human, and to drop off completed work at the Inventory Station. For pickup, the robot moves its end effector near to the human's right hand so that the human can hand over her work. To distinguish between $\mathcal{S}$ and $\neg\mathcal{S}$ during dropoff, as discussed in Sec. 3.3.2, we use a force sensor mounted at the end effector to judge whether or not the object successfully made contact with the counter.

3.4.2 Controller Implementation

The decision tree produced by Slugs serves as an online look-up table during robot operation. After transitioning to the next commanded state, the HSR will update its knowledge of the environment (mainly, any obstacles, if a dropoff action was successful, and the human's current BL). We used SMACH[3] to implement a finite-state machine framework that bridges the gap between the high-level action policy from Slugs and the robot's lower-level sequential task executors.

Once the Slugs planner determines the next desired action, the robot's required skills are executed sequentially by the sub-task controller that is associated with a particular SMACH state. In other words, the sub-task control layer is responsible for decomposing the desired Slugs action into the sequential, lower-level required skills. For example, when the robot has completed work, the action Go_{S0}

[3]http://wiki.ros.org/smach

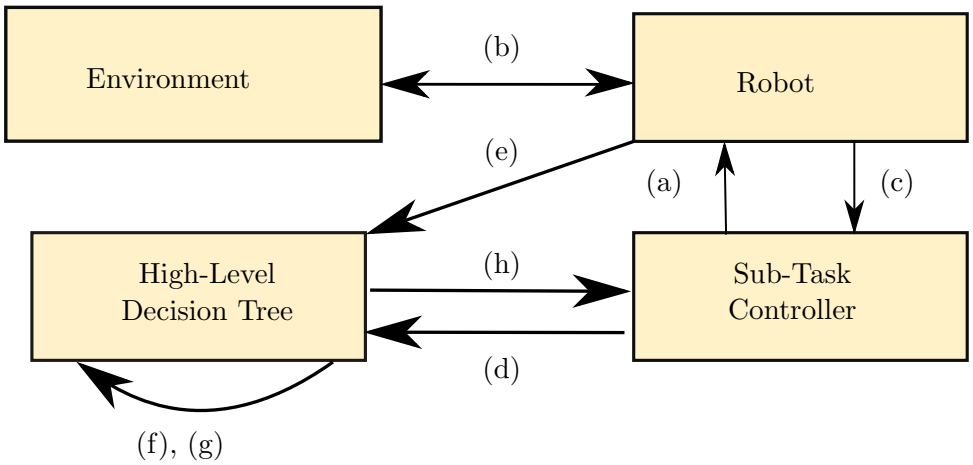

Figure 3.5: The communication protocols between the the robot and the sub-task (SMACH) and high-level (Slugs) controllers. The systems communicate by repeating (a)-(h), where: (a) Track and command sequential tasks; (b) Perceive, navigate, and manipulate; (c) Sequential tasks complete; (d) Request next action; (e) Subscribe to ROS environment topics; (f) Check for human at workstation. Update RS and BL when worker is present; (g) Select next action from look-up table. (h) Respond with next action.

first contains navigation (move to counter at State 0), then manipulation (drop off object), and finally navigation (move back from counter) skills. The sequence of behaviors thus requires recognition of when the previous sub-task succeeds. All robot sub-tasks are programmed with the structure of ROS actions in order to be flexible to sub-task execution times. Once a high-level action is completed, or the first dropoff try is unsuccessful, SMACH requests the next desired action from the Slugs planner. The system architecture is shown in Fig. 3.5.

3.4.3 Results

Through experimentation, we sought to verify that our automatically-synthesized, high-level controller properly reacts to its environment while maintaining (3.15). Fig. 3.6 shows the layout of our experimental setup and the positions of States 0 through 3. Referring to the BL model in (3.1), we take $t_d = 10s$.

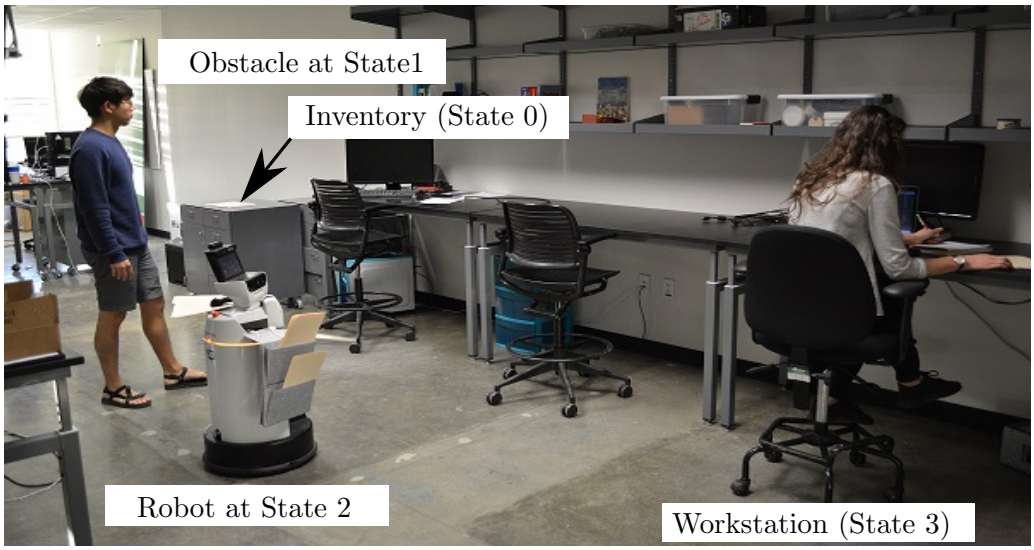

Figure 3.6: Experimental setup with four positions in the workspace, corresponding to $RS = 0$ (Inventory Station), 1, 2, and 3 (Human Workstation). The HSR transports deliverables in its satchel and carries completed work in its manipulator back to the Inventory Station. A human obstructs its path, causing it to remain in State 2 until the next time step, by which time the human will have departed.

To test the robustness of the controller, we allowed the HSR to operate autonomously for 30 minutes. During this time, the robot reacted to obstacles, interacted with the worker at the workstation, and returned to the Inventory Station several times to drop off completed work, as shown in Fig 3.7. While we did not account for the human taking a break in our reactive synthesis problem, we incorporated this consideration for our experiments. If the HSR does not sense a human at the workstation, the HSR waits until she reappears, and then proceeds with its actions according to the Slugs planner.

It is also of interest to investigate the high-level controller behavior for differing BL_{init} values. Fig. 3.8 shows the results with $BL_{init} = 30\%$ ($BL_{init} = \frac{9}{30}$ in Slugs).[4] We also highlight whether the robot is manipulating completed work, and the implementation of the work dropoff logic at State 0. Fig. 3.9 shows the

[4]This experiment is shown at https://youtu.be/My6WIiZZCsM.

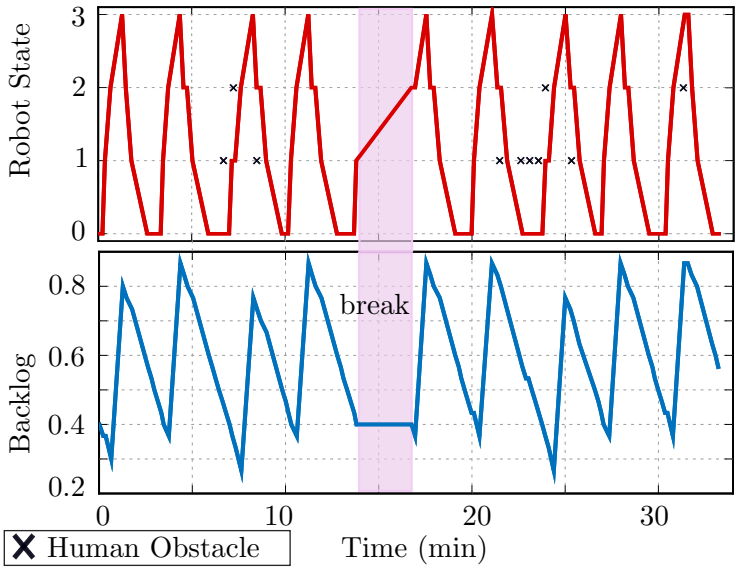

Figure 3.7: Robustness test with $BL_{init} = 40\%$. Human obstacles that appear in States 1 and 2 and depart by the next time step are marked by black x's. The worker moves out of the workspace for three minutes, which is highlighted in pink. The robot waits for the human to return at $RS = 2$, but does not update RS and the environment variables in the Slugs planner until the human returns to work.

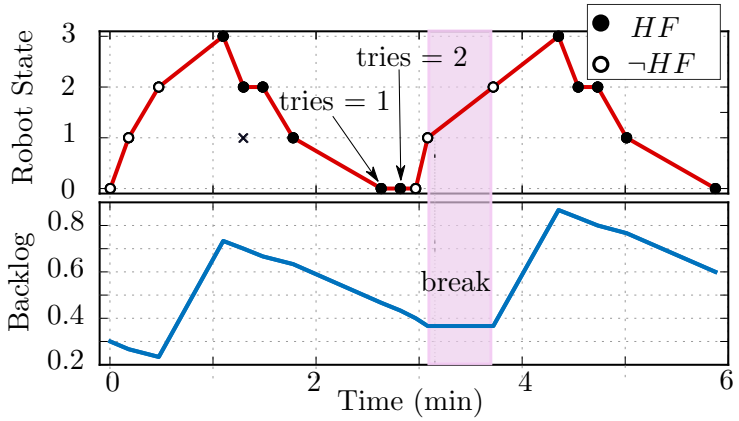

Figure 3.8: Robot behavior with $BL_{init} = 30\%$. When $RS = 1$ and HF at 1.75 minutes, the SMACH sub-task controller tracks the amount of time the robot takes to travel from State 1 to State 0 and execute the dropoff sub-tasks, during which time, tries = 1. If this time exceeds 35 seconds, $\neg\mathcal{S}$ is communicated to the high-level controller. The Slugs planner updates the tries value to be equal to 2, and commands that the robot continue to execute its dropoff sub-tasks in the next time step. After the second try, the robot successfully drops off the work, and the robot's manipulator is empty again.

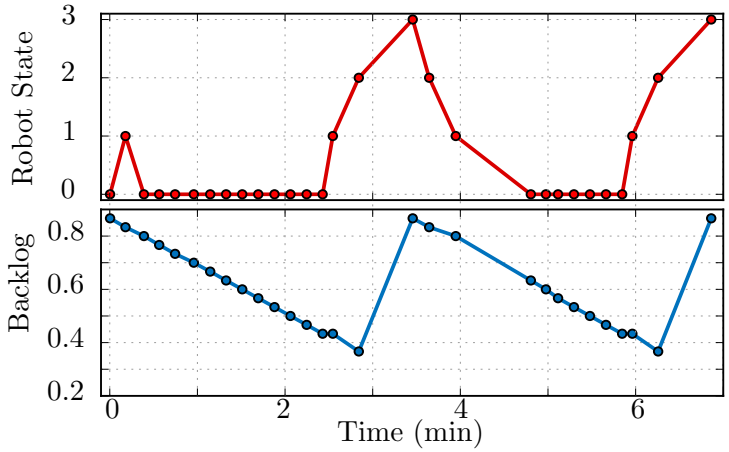

Figure 3.9: Robot behavior with $BL_{init} = 86.7\%$. The circular markers indicate the times at which the sub-task controller calls the high-level decision tree to request the next robot action. Excluding its initial movement to and from State 1, the robot elects to wait at State 0 until it moves to deliver work, even when there are no obstacles present.

robot's behavior with $BL_{init} = 86.7\%$ ($BL_{init} = \frac{26}{30}$ in Slugs). We note that the robot first moves to State 1, moves back to State 0, and remains at this position until it proceeds to the workstation to provide deliverables. The initial movement to and from State 1 upholds a winning strategy in the two-player game, but provides no useful output. Additional specifications could be imposed in the reactive synthesis problem to incorporate robot energy efficiency as a consideration to eliminate unnecessary robot movements.

3.5 Discussion

In all three experiments, the robot successfully maintains human BL needs and can autonomously reason about and react to its environment. In the situations considered, the BL value reaches 86.7%, but does not exceed 87%, as desired. It is interesting to note that the controller strategies drive the robot to wait in State 0 until it moves to deliver new work. The robot does not wait in States 1 or 2 for extended periods of time. This behavior is sufficient to satisfy system specifications, but optimizing where the robot waits could be needed in other work environments. For the experimental setup considered, we have verified that our proposed controller offers robustness and flexibility for a real-time, human-centered application.

There are several interesting areas remaining for future work. The human backlog model in the presented case study is a toy model used for proof-of-concept. In the future, we are interested to incorporate verified, dynamic human models based on psychology and cognitive theory, to consider how factors such as fatigue, training, motivation, and stress impact backlog. We are also interested to study how we may extend our work to produce not only feasible motions, but also optimal motions. For now, our results support the feasibility of designing robots that are formally guaranteed to reduce the human share of work activities in the execution of everyday tasks.

Chapter 4

Conclusions

This work examines two critical considerations in devising human-centered robots: producing optimal trajectories that are dynamically consistent with low-level robot components, and synthesizing high-level, human-aware robot controllers. It is interesting to note that in one study, we concentrate on generating optimal motions, while in the other we focus on the realizability of robot controllers, without any optimization considerations. The differing approaches both give insight on the capabilities of robot trajectory planners and controllers. What these two studies have in common is the consideration of limitations that need to be taken into account for robot planning: in one case, actuator state and input constraints; in the other case, the productivity abilities of humans.

In Chapter 2 we explore how to efficiently incorporate SEA dynamics into our planning scheme, based on sequential linear optimization. Our experimental results demonstrate the benefit of our approach, which leverages actuator compliance in executing high speed motions. As actuators cannot function as perfect torque sources, planners that have the knowledge of the actuators' more-detailed abilities will allow them to produce achievable trajectories which can leverage the natural dynamics endowed by their low-level components.

Human factors are often neglected in formal methods when devising robot controllers, at the expense of having no formal guarantees that the robot can actually realize a human-centered goal. In Chapter 3, we leverage reactive synthesis to devise a controller that supports a human's and a robot's collaborative goal. By formulating a work delivery scenario as a two-player game, we automatically synthesize a controller that considers a human's work needs and supports robustness to system disturbances. Experimental validation on the HSR hardware demonstrates that the high-level controller strategy enables the robot to operate autonomously and robustly, while considering the activities and productivity of the human worker.

Not only does robot planning need to consider the mechanical abilities of the low-level robotic components, but also the abilities of the humans that the robots seek to assist. This thesis demonstrates the potential to devise efficient, robust, fit-for-purpose planners and controllers that promote the overarching goal of devising robots that that can collaborate with and support humans in the execution of a variety of tasks.

Appendix A

Equilibrium Nominal Trajectory

We seek to identify the equilibrium actuator states, x_{eq} and the corresponding input currents, u_{eq}, for the Draco P1 leg. In this case there are 10 unknowns: eight actuator states and two input currents. When the system is in equilibrium, the A_{lin}, B_{lin}, and $bias$ matrices do not change.

We must linearize about the initial desired configuration of the Draco robot in order to obtain the A_{lin}, B_{lin}, and $bias$ terms. With the initial desired joint configurations known, we can calculated the corresponding initial actuator lengths, $z_{1,init}$ and $z_{2,init}$. We can then formulate a nominal actuator state vector, $x_{nom} = \begin{bmatrix} 0 & 0 & z_{1,init} & 0 & 0 & 0 & z_{2,init} & 0 \end{bmatrix}^T$

While x_{nom}, which is used for linearization, corresponds to the correct initial robot configuration, we must find the initial condition that is at equilibrium with the robot's springs. Considering (2.20), at equilibrium, $x_{eq} = x_{n+1} = x_n$, and the following holds:

$$x_{eq} = A_{lin}x_{eq} + B_{lin}u_{eq} + bias. \tag{A.1}$$

Simplifying this equation gives:

$$(I - A)x_{eq} = B_{lin}u_{eq} + bias \tag{A.2}$$

This expression provides eight equations. Two more equations are needed so that there are as many equations as unknowns. We leverage that the initial actuator lengths are known and use the equations that enforce that $\delta_{init} + y_{init} = z_{init}$:

$$S_z x_{eq} = \begin{bmatrix} z_{1,init} \\ z_{2,init} \end{bmatrix}^T \tag{A.3}$$

where the selector matrix, S_z, captures the displacement terms for the 2-link system:

$$S_z = I_p \otimes \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}. \tag{A.4}$$

We can now formulate a nonsingular matrix to solve for x_{eq} and u_{eq}:

$$\begin{bmatrix} (I - A_{lin}) & -B_{lin} \\ S_z & 0 \end{bmatrix} \begin{bmatrix} x_{eq} \\ u_{eq} \end{bmatrix} \triangleq \mathcal{A}_{equ} \begin{bmatrix} x_{eq} \\ u_{eq} \end{bmatrix} = \begin{bmatrix} bias_{lin} \\ z_{1,init} \\ z_{2,init} \end{bmatrix} \tag{A.5}$$

$$\begin{bmatrix} x_{eq} \\ u_{eq} \end{bmatrix} = \mathcal{A}_{equ}^{-1} \begin{bmatrix} bias_{lin} \\ z_{1,init} \\ z_{2,init} \end{bmatrix} \tag{A.6}$$

We can now formulate the initial trajectory, $\mathbf{X}$, that is used for linearization during the first iteration of our strategy. The actuator states of $\mathbf{X}$ at each time step are equal to x_{eq}. Additionally, the values of u_{eq} can be used to form the baseline control signal at each time step for $\mathbf{U}_{baseline}$. $\mathbf{U}_{baseline}$ can then be used in the cost function to penalize input currents that deviate from this signal.

Bibliography

[1] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1995, pp. 399–406.

[2] N. Paine, S. Oh, and L. Sentis, "Design and control considerations for high-performance series elastic actuators," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 1080–1091, 2014.

[3] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle, "A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on MABEL," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1170–1193, 2011.

[4] R. Schlossman, G. C. Thomas, O. Campbell, and L. Sentis, "Exploiting the natural dynamics of series elastic robots by actuator-centered sequential linear programming," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 1405–1412.

[5] J. Pratt, T. Koolen, T. De Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomo-

tion, part 2: Application to m2v2, a lower-body humanoid," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1117–1133, 2012.

[6] B. Vanderborght, A. Albu-Schäffer, A. Bicchi, E. Burdet, D. G. Caldwell, R. Carloni, M. Catalano, O. Eiberger, W. Friedl, G. Ganesh *et al.*, "Variable impedance actuators: A review," *Robotics and autonomous systems*, vol. 61, no. 12, pp. 1601–1614, 2013.

[7] L. Chen, M. Garabini, M. Laffranchi, N. Kashiri, N. G. Tsagarakis, A. Bicchi, and D. G. Caldwell, "Optimal control for maximizing velocity of the compact compliant actuator," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on.* IEEE, 2013, pp. 516–522.

[8] A. Radulescu, M. Howard, D. J. Braun, and S. Vijayakumar, "Exploiting variable physical damping in rapid movement tasks," in *Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on.* IEEE, 2012, pp. 141–148.

[9] D. Braun, M. Howard, and S. Vijayakumar, "Optimal variable stiffness control: formulation and application to explosive movement tasks," *Autonomous Robots*, vol. 33, no. 3, pp. 237–253, 2012.

[10] D. J. Braun, F. Petit, F. Huber, S. Haddadin, P. Van Der Smagt, A. Albu-Schäffer, and S. Vijayakumar, "Robots driven by compliant actuators: Optimal control under actuation constraints," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1085–1101, 2013.

[11] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle, "Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on MABEL," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 324–345, 2013.

[12] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1447–1454.

[13] F. L. Moro, N. G. Tsagarakis, and D. G. Caldwell, "A human-like walking for the COmpliant huMANoid COMAN based on CoM trajectory reconstruction from kinematic motion primitives," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 364–370.

[14] A. Werner, B. Henze, F. C. Loeffl, S. Leyendecker, and C. Ott, "Optimal and robust walking using intrinsic properties of a series-elastic robot," in *IEEE-RAS International Conference on Humanoid Robots*, 2017.

[15] A. Werner, W. Turlej, and C. Ott, "Generation of locomotion trajectories for series elastic and viscoelastic bipedal robots," in *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[16] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[17] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1, 2013, pp. 1–10.

[18] V. L. Orekhov, C. S. Knabe, M. A. Hopkins, and D. W. Hong, "An unlumped model for linear series elastic actuators with ball screw drives," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 2224–2230.

[19] S. Schütz, A. Nejadfard, C. Kötting, and K. Berns, "An intuitive and comprehensive two-load model for series elastic actuators," in *Advanced Motion*

Control (AMC), 2016 IEEE 14th International Workshop on. IEEE, 2016, pp. 573–580.

[20] H. Asada and J. Leonard, *2.12 Introduction to Robotics.* Fall Massachusetts Institute of Technology: MIT OpenCourseWare, License: Creative Commons BY-NC-SA, 2005. [Online]. Available: https://ocw.mit.edu

[21] R. Schlossman, https://github.com/rschloss123, 2018.

[22] T. Koolen, S. Bertrand, G. Thomas, T. De Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot atlas," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1650007, 2016.

[23] G. C. Thomas and L. Sentis, "Towards computationally efficient planning of dynamic multi-contact locomotion," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.* IEEE, 2016, pp. 3879–3886.

[24] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014.

[25] S. H. Kwak and S. Oh, "Comparison of resonance ratio control and inner force control for series elastic actuator," in *Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE.* IEEE, 2017, pp. 7583–7588.

[26] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[27] J. Sutherland and B. Bennett, "The seven deadly wastes of logistics: applying toyota production system principles to create logistics value," *White paper*, vol. 701, pp. 40–50, 2007.

[28] R. Schlossman, M. Kim, U. Topcu, and L. Sentis, "Toward achieving formal guarantees for human-aware controllers in human-robot interactions," *arXiv preprint arXiv:1903.01350*, 2019.

[29] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[30] S. Maniatopoulos, P. Schillinger, V. Pong, D. C. Conner, and H. Kress-Gazit, "Reactive high-level behavior synthesis for an atlas humanoid robot," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4192–4199.

[31] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Reactive synthesis for finite tasks under resource constraints," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 5326–5332.

[32] Y. Zhao, U. Topcu, and L. Sentis, "High-level planner synthesis for whole-body locomotion in unstructured environments," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 6557–6564.

[33] Y. Wang, N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "Task and motion policy synthesis as liveness games," in *ICAPS*, 2016, p. 536.

[34] A. Ramachandran, C.-M. Huang, and B. Scassellati, "Give me a break!: Personalized timing strategies to promote learning in robot-child tutoring," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2017, pp. 146–155.

[35] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Planning with trust for human-robot collaboration," in *Proceedings of the 2018 ACM/IEEE*

International Conference on Human-Robot Interaction. ACM, 2018, pp. 307–315.

[36] B. Wu, B. Hu, and H. Lin, "Toward efficient manufacturing systems: A trust based human robot collaboration," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 1536–1541.

[37] M. Webster, C. Dixon, M. Fisher, M. Salem, J. Saunders, K. L. Koay, K. Dautenhahn, and J. Saez-Pons, "Toward reliable autonomous robotic assistants through formal verification: a case study," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 2, pp. 186–196, 2016.

[38] M. Kwiatkowska, "Cognitive reasoning and trust in human-robot interactions," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2017, pp. 3–11.

[39] D. Porfirio, A. Sauppé, A. Albarghouthi, and B. Mutlu, "Authoring and verifying human-robot interactions," in *The 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, 2018, pp. 75–86.

[40] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 66–73.

[41] M. Rausch, A. Fawaz, K. Keefe, and W. H. Sanders, "Modeling humans: A general agent model for the evaluation of security," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2018, pp. 373–388.

[42] L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu, "Synthesis of human-in-the-loop control protocols for autonomous systems," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 450–462, 2016.

[43] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[44] S. J. Jorgensen, O. Campbell, T. Llado, D. Kim, J. Ahn, and L. Sentis, "Exploring model predictive control to generate optimal control policies for hri dynamical systems," *arXiv preprint arXiv:1701.03839*, 2017.

[45] J. Heard, C. E. Harriott, and J. A. Adams, "A survey of workload assessment algorithms," *IEEE Transactions on Human-Machine Systems*, no. 99, pp. 1–18, 2018.

[46] W. MacDonald, "The impact of job demands and workload on stress and fatigue," *Australian Psychologist*, vol. 38, no. 2, pp. 102–117, 2003.

[47] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[48] R. Ehlers and V. Raman, "Slugs: Extensible gr (1) synthesis," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 333–339.