Simulation Based Fault Detection and Design Modification for Highly Dynamic Robotic Systems

Samuel Robert Zapolsky

April 11, 2017

B.A. in Economics & International Affairs, May 2012, George Washington University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

May 21, 2017

Dissertation directed by

Evan M. Drumwright
Toyota Research Institute

The School of Engineering and Applied Science of The George Washington University certifies that Samuel Robert Zapolsky has passed the Final Examination for the degree of Doctor of Philosophy as of February 24, 2017. This is the final and approved form of the dissertation.

Simulation Based
Fault Detection and Design Modification
for Highly Dynamic Robotic Systems

Samuel Robert Zapolsky

Dissertation Research Committee:

Evan Drumwright, Senior Research Scientist, Toyota Research Institute, Dissertation Director

Gabriel Parmer, Department of Computer Science, George Washington University, Committee Chair

Andy Ruina, Professor of Mechanical Engineering, Cornell University, Committee Member

Stefan Schaal, Professor of Computer Science, Neuroscience, and Biomedical Engineering, University of Southern California, Committee Member

Aaron Johnson, Professor of Mechanical Engineering, Carnegie Mellon University, Committee Member

Rahul Simha, Professor of Computer Science, George Washington University, Committee Member

In memory of my good friends Brock and Navdeep

and

For my parents Valerie and Jeffrey and my brothers Benjamin, Nathan, and Ivan

Acknowledgments

I would first like to thank my thesis advisor Dr. Evan Drumwright who is now a Senior Research Scientist at Toyota Research Institute. When I started my work in robotics, I was completing a bachelor's degree in international affairs and had very few tangible qualifications, just a will to succeed and a desire to learn. Prof. Drumwright's confidence in and willingness to vouch for my potential might have been the single, pivotal reason behind how I was able to break into the field of robotics. His continual support has enabled me to succeed and has demonstrated the type of mentorship that I believe all advisors should aspire to.

I would also like to thank by fellow researchers at the Italian Institute of Technology (IIT) in Genoa, Italy: my lab mates Hamza Khan, Michele Focchi, Victor Barasuol, Ioannis Havoutis, Jake Goldsmith, Thiago Bonaventura, Marco Frigerio and our mentors Jonas Buchli and Claudio Semini. The researchers at IIT contributed to my first published paper and introduced me to the subject of legged locomotion in robotics. Without their passionate participation in my research and genuine friendship I would not be the researcher I am today.

I have mentored or participated in mentoring several students throughout my doctoral study: Matthew Scaperoth, Joshua Shapiro, Jonathon Shepherd, and Bradley Canaday to name only a few. I hope their time in the Positronics Lab and their experience with collaborative coding and research was as education to them as it was for me.

Finally, I must express my gratitude to my lab mates in the Positronics Lab at GWU: James Taylor, Roxana Leontie and Joshua Lurz for providing me with unfailing support and continuous encouragement throughout my years of study and throughout the process of researching and writing this thesis. I wish them the best in completing their thesis; I hope I can continue to support you all as you have supported and helped me grow through the years. This accomplishment would not have been possible without them. Thank you.

Abstract

Simulation Based
Fault Detection and Design Modification
for Highly Dynamic Robotic Systems

The current approach to creating and improving robots and their control systems follows a cycle where complex mechanisms and controllers are iteratively designed, briefly debugged in a simulated environment, built, tested and then redesigned to address anomalous behaviors that were observed during *in situ* testing. During the initial cycles of this procedure, when unvetted control systems are tested on physical hardware, easily avoidable errors (e.g., unexpected collisions from inexact link geometry measurements or parasitic oscillation in actuators and passive elements) can have catastrophic consequences.

Although testing robots *in situ* can be costly, often entirely new phenomena emerge from situated testing that were not observed throughout early computerized simulation-based testing. The discovery of such unanticipated behavior is currently considered a normal occurrence during robotics testing *in situ*. This anticipated unanticipated behavior—the expectance of the unknown during robot operation—is due in part to the increased complexity of robotic systems (e.g., uneven terrain, impacting collision, unpredictable contact conditions) compared to typical automotive or aerospace applications that rely heavily on simulation-based testing before physical testing.

I have investigated a statistical approach to simulation, where the indeterminacy of physical models or uncertainty in the structure of a mechanism or its environment is represented as a collection of particles in many parallel simulations. The aim of this approach is to inform roboticists of the possible unknown or unexpected behaviors that a robotic system may exhibit in order to address these faults before they have been observed on the physical system. Our approach excites many of the errors caused by modeling, sensory, actuation, and communication error or uncertainty and can assist roboticists in determining whether certain robot designs, control systems, or modeling assumptions might result in hard-to-predict behavior; I use this information to predict the robustness or brittleness control policies and then validate the predictions *in situ*, on low-cost quadrupedal robots.

This dissertation presents tools for automating and or simplifying a roboticist's typical workflow (i.e., designing, testing, controlling, and debugging robots). These tools aim to inform roboticists of the possible unknown or unexpected behaviors that a robotic system may exhibit in order to address these faults before they have been observed on the physical system. The goal of this dissertation is to greatly accelerate the design-build-test cycle of research in robotics by providing a readily usable virtual testing and design framework for robot hardware and software.

Table of Contents

List of Figures

List of Tables

1 Introduction

A roboticist begins the process of designing robotic hardware and software by identifying a task to automate and then conceiving of a concept for the robotic platform or control system that will perform that task. For designing robotic hardware, this concept includes a relative size (e.g., nanometer, centimeter, meter), and a morphological type (manipulator, quadruped, biped, wheeled). The designer uses intuition, experience, biological inspiration, or some combination of the three to design the robot (i.e., by selecting the kinematic, dynamic, and geometric parameters of the robot model). The designer then uses some computational tools (e.g., finite element analysis, rigid body dynamics simulation, MATLAB's SIMULINK) to assess the feasibility of the robot model by conducting preliminary checks to see whether the robot can satisfy its targeted functionality. Once the designer is confident in the model design, the robot can be fabricated for *in situ* testing. Subject to performance on sample tasks, the designer will iteratively adjust the physical parameters of the robot, presumably until performance targets are met. If instead a roboticist uses an existing hardware platform, then a similar formula is followed to test and debug perception, planning, and control software: the roboticist will identify a problem, design and develop software, and run tests and experiments to assess the performance of the software on a physical robot, and repeat.

Safely carrying out this trial and error-based robot development *in situ* is tedious and time consuming. Simulation-based testing for robot design and control system verification can conceivably provide an effective intermediate phase between the design of a robotic system and testing *in situ*. Automotive and aerospace industries utilize computer-aided engineering tools as well as *in situ* unit testing to ensure that a physical system will work as expected when the final system is built and tested. While unit testing for hardware is common in robotics, the use of simulation for testing in general robotics applications has not been particularly reliable because of the complexity of robotic systems and their environments make simultaneously fast and accurate simulation tools difficult to develop.

How can simulation be improved to mitigate the problems of unpredictability and model infidelity on complex robotic systems? One possibility would be to increase the accuracy of simulation through use of higher fidelity models, though such models generally require increased computation and considerable data collection for system identification. Even so, a carefully tuned model is very unlikely to exactly match its physical counterpart. Predictions made in simulation do not typically match reality—a physically situated robot can exhibit behaviors that substantially diverge from expectation. Such anomalies can lead to unpredictable robot behavior and unexpected failures that may damage the robot or its surroundings; this unpredictability precludes extensive testing of a robotic system *in situ* (and inhibits robots from entering human-populated environments). A second possibility would add noise to different aspects of a simulated robot (e.g., external force perturbation, injected sensor noise) in an attempt to account for infidelities. However, randomly added noise will not necessarily improve the predictive power of a simulation: adding bias to a virtual sensor may add uncertainty to a simulation, but unless the bias is representative, predictability will not improve.

How can simulation be improved to encourage roboticists to adopt virtual testing and design tools into their workflows? This thesis argues for a third option: modifying simulation to incorporate the uncertainty resulting from model infidelity and system identification error, toward identifying gross behavioral defects that could result from that uncertainty—essentially a parallelized version of the second option. Predicting the divergent behavior that a robot might exhibit with simulation-based testing offers an alternative to accurately predicting the behavior of a robot *in situ*.

1.1 Designing and testing robotic systems subject to uncertainty

This dissertation will show how a simulation-based testing approach can serve as a tool for virtual robotic prototyping and testing. I describe a Monte Carlo method approach to simulation-based testing that incorporates uncertainty into a robot's kinematic, dynamic, and geometric models to explore the range of possible behaviors that the physical robot may exhibit when operating under a given control policy within a particular environment. The approach excites and then interprets

the output of robotic system behaviors that result from errors in system identification and state estimation. I then present a suite of robot design modification tools toward mitigating the occurrence of anomalous behaviors or diminishing their negative effects once detected. I address the primary theoretical and practical challenges toward implementing the described virtual testing and design systems. I also provide (1) a description of an implementation of a Monte Carlo method framework toward simulating many perturbed versions of a target robotic system; (2) experimental evidence supporting my hypothesis that simulation can be used to predict and correct for failing or divergent behavior of the targeted robotic system *in situ*; and (3) the algorithms necessary to realize real-time simulation of complex robotic systems in order to facilitate (1) and (2).

Although the concept of randomly sampling external and internal influences on systems to gauge their behavior under non-ideal circumstances is not new—see e.g., Monte Carlo methods (Abbas et al., 2013), sensor noise modeling (Ponton et al., 2016), control signal perturbation (Bemporad & Morari, 2007), external force perturbation (Hutter et al., 2014)— there remain theoretical and practical challenges to implementing such an approach for robotics applications: accurately modeling intermittent contact; ensuring that evaluations of classical mechanics formulae are computationally fast (efficient) but remain stable and accurate to theoretical models; gauging which theoretical models are true to reality and when alternative models should be used; automating the analysis of large volumes of data to produce useful feedback to a roboticist; and developing the interactive tools that make such a system easy to incorporate into the existing paradigms in robotics (e.g., robot morphological design, controller falsification, model predictive control). This thesis describes how these obstacles may be overcome and presents results from the software implemented toward realizing computer-aided tools for robot-oriented testing and design.

1.2 Contributions and organization of chapters

The next chapter, Chapter 2, covers related works in both multi-rigid body simulation and robotics that address topics similar to those in this thesis. That chapter also provides necessary background information and theory that will be drawn upon throughout following chapters. The

work in this thesis builds upon work in physical simulation, robust planning and control, validation and physical testing, automated robot design, evolutionary robotics, and the planning and control of locomotion and manipulation for limbed robots. I seek to extend such work by providing a framework for rapid falsification of brittle control policies (Chapters 3 and 4) and modifying a robot's construction to improve robustness (Chapters 5 and 6).

Chapter 3 describes the *particle traces* approach to simulated testing. Particle traces, also referred to as "ensembled" trajectories (Mordatch et al., 2015), refer to the output of a Monte Carlo-method-driven approach to simulating robotic systems subject to uncertainty. The sampling-based virtual testing approach presented in Chapter 3 perturbs robot modeling parameters, sensory readings, and actuator commands to evaluate a robot's behavior statistically over a range of inputs and environments. The research described in Chapter 3 has worked toward understanding how multi-rigid body simulations can better characterize the behavior of imperfectly modeled robots subject to sensing and modeling uncertainty. When contact is expected but does not occur or when contact is not expected but does occur, robot behavior is likely to diverge from plan, often drastically. When the occurrence of such events or lack thereof is dependent on only small perturbations to the state of the robotic system, they are referred to as *grazing bifurcations*. Correspondingly, Chapter 3 describes an approach that uses simulation to detect possible such behavioral divergences on real robots. This approach, and others like it, could be applied to validation of robot behaviors (Chapter 4), mechanism design (Chapters 5 and 6), and even online planning (Section 4.3.3). The presented approach seeks to bridge the extremes of isolated physical simulation tests and full-on testing on real robotic hardware and is straightforward to describe, easily implemented, and uses techniques already familiar to many roboticists.

Chapter 4 presents experiments testing and validating the performance of the particle traces approach detailed in Chapter 3; it provides demonstrations of a virtual falsification approach to simulation-based testing. Falsification aims to identify points of failure in planning and control software as well as brittle regions of a robot's parameter space. I demonstrate that combining even coarse estimates of state and robot morphological parameters with fast multi-rigid body simulation

can be sufficient both to detect failure-inducing divergent robot behavior and to characterize robot performance in the real world. Given the extensive data that such simulations are capable of generating, this approach could be used to assess risk and find and analyze likely failures. I assess this falsification approach on both actuated, high degree-of-freedom robot locomotion examples and a picking task with a fixed-base manipulator. The chapter concludes by demonstrating an implementation of the particle traces approach for online planning; I use this implementation to decide the best timing for abruptly switching between two locomotion control policies. As a display of the flexibility of the approach, I show that particle traces can also be used as a system stability indicator which, while "weaker" than a proof, is readily applicable to the challenging problem of analyzing high dimensional systems that undergo nonsmooth behavior.

Chapter 5 presents an algorithmic approach toward mitigating the divergent behaviors exhibited by the robots in Chapter 4; it presents a framework in which to model the various factors that limit the best-case capabilities of a robot. The limitations in this chapter often define the robot's proximity to a grazing bifurcation detected through the sampling approach presented in Chapter 3. I describe a method for using witness functions that represent a robot's proximity to exceeding limitations encountered during virtual control of the robot in order to improve its model within simulation. The presented virtual prototyping process explores the space of robot morphological parameters to determine whether a robot's hardware will be able to perform a task even under ideal conditions (e.g., if balance were not an issue for a locomoting robot). If the modeled robot is incapable of performing a target task, the virtual testing phase can determine which of the robot's morphological parameters limits its peak performance and how the robot might be modified to overcome this limit. This approach can be used to maximize a robot's performance on a task given known failure conditions (like the divergent behavior-inducing grazing bifurcation described in Chapter 3). I also present an algorithm in this chapter for updating a robot's morphological parameters to improve the robot's ability to perform its target task or tasks.

Chapter 6 validates the virtual prototyping process from Chapter 5 by demonstrating how predicted improvements made in the presented framework can translate to improved task performance

and reduced risk of failure during physically situated testing. I show how this approach can interactively provide a robot designer feedback on which morphological parameters are likely to limit the performance of a robot and how to modify the design to remove or mitigate such limitations. Through simulated prototyping and testing, I validate the model updating process by improving a coarsely fabricated robot, built with 3D printed parts and low-cost actuators. This robot is initially incapable of performing a walking gait due to hardware limitations and breaks after few trials; the improved robot demonstrates durability and successful behavior after the modifications from simulated prototyping are applied. I further validate the model updating approach in a second example that reconfigures a robot with adjustable limb length and body size toward improving locomotion capability. The demonstrations in this chapter show how engineers can update a robot's design and iteratively adjust its morphological parameters to make efficient use of available hardware.

Chapter 7 presents inverse dynamics controller formulations toward providing robots with accurate control in the presence of contact; it describes the theory behind producing contact force predictions for inverse dynamics problems that are accurate to the rigid contact model. The algorithms presented in this chapter make possible the more stable approach for faster, accurate simulation presented in Chapter 8. In the context of robot control, using inverse dynamics requires knowledge of all forces, including contact forces, acting on the robot. Existing such inverse dynamics approaches have used approximations to accepted contact models to permit the use of fast numerical linear algebra algorithms. In contrast, I describe inverse dynamics algorithms that are derived only from first principles and use established phenomenological models like Coulomb friction. I assess these inverse dynamics algorithms against both error feedback control and inverse dynamics control with virtual contact force sensing. I demonstrate that a range of inverse dynamics controllers offer different advantages depending on the target task and robotic system. Some inverse dynamics algorithms offer a boost in tracking accuracy in exchange for computation time; other inverse dynamics algorithms offer fast computation but require strict conditions to be satisfied to operate (e.g., non-redundant contact) or can make unrealistic assumptions about contact conditions (e.g., always sticking friction). The chapter concludes by overviewing the advantages

and disadvantages of various inverse dynamics algorithms to guide a roboticist toward selecting the best algorithm for their application.

Chapter 8 describes how to improve the numerical stability and thus simulation speed for robots controlled with error feedback toward realizing real-time simulation of the complex robotic systems employed in this thesis. Roboticists modeling control of manipulator and legged robots often assume force/torque-based control using an open-loop model of voltage, hydraulic pressure, or pneumatic pressure to actuator torques. This chapter shows that such force/torque-based models can lead to stiff differential equations, which are computationally inefficient to solve: relatively small integration steps are necessary to ensure stability. I have investigated two approaches which appear to mitigate this problem: incorporating transmission modeling (applicable only to electromagnetic actuators at present) and inverse dynamics control (addressed in Chapter 7). Both of these approaches require increased computation per integration step, but this chapter will demonstrate that the larger integration steps can still yield considerably higher simulation throughput. This chapter also identifies research challenges with using inverse dynamics control within multi-rigid body simulations. In particular, I examine the state of the art approach for integrating the simulation subject to contact and inverse dynamics constraints, and I identify algorithmic challenges, both theoretical and practical.

Chapter 9 describes the online locomotion planning and control software used within the experiments in this dissertation. A definitive reference on programming a locomotion and control system has been unavailable, as a "best" method of controlling a locomoting robot is an open problem. The approach described in this chapter aims to reduce the control inputs into the locomotion system to planar commands, similar to those of a car. My implementation of a locomotion planner for a quadrupedal robot will provide insight into the context of the design decisions made in this work. This chapter also presents a software foundation on top of which the more complex simulation tools presented in this work are implemented.

I conclude the thesis in Chapter 10 with a discussion of the conclusions drawn from the experimentation described in this work. This chapter discusses the implications of this work toward how

roboticists can use simulation to produce useful predictions for poorly modeled physical systems. This thesis presents and validates two distinct contributions: (1) a method of detecting brittle control policies and robotic system parameters and (2) a modification strategy for mitigating those detected failures. This chapter discusses how such tools will continue to shape robotic experimentation as the need for rapid development of robots drives the virtualization of testing processes, and the ubiquity of robots increases the need for interactive and user-friendly robot design tools.

2 Background and Related work

This chapter covers related work in both simulation and robotics that address topics similar to those this thesis. The work in this thesis builds upon work in robust planning and control (§2.1), automated robot design (§2.2), validation and physical testing (§2.2.4), the planning and control of limbed locomoting and manipulating robots (§2.3), and physical simulation (§2.5). This dissertation aims to extend the work referred to in this chapter by providing a simulation-based tool for control policy fault detection and an interactive robot design suite aimed at improving a robot's morphology toward robust, successful designs.

2.1 Planning with uncertainty

Trajectory planning with imperfect state makes the problem of finding the best path through an environment more difficult. Roboticist's often attempt to find ways of faithfully representing the uncertainty in state of a controlled system, and then integrating that system's expected state and its uncertainty over time as the system is controlled. The tradeoff in representing such systems is one between the complexity of the representation and its evaluation, versus introducing a more simplified abstraction of the system's uncertainty without excessive loss. Prentice & Roy (2009) propose a computationally efficient method of factoring the covariance matrix representing Gaussian noise on a linear system to combine several prediction and measurement steps subject to uncertainty into a single operation; robotic systems subject to measurement and control uncertainty are commonly represented as linear system with Gaussian noise for robust planning application (van den Berg et al., 2011; Kewlani et al., 2009). Rather than using bounded Gaussian uncertainty (e.g., at three standard deviations from the expected mean state), Marruedo et al. (2002) assume a white noise additive uncertainty at each process model iteration, but with Lipschitz continuity bounding the evolution of the system's state. The linear Gaussian/additive uncertainty method assumes that the evolution of the system over time can be approximately represented as unimodal. More complex representations must be used when assuming that dynamical systems might exhibit bifurcation (i.e., state divergence between systems of similar parameterization and initial state).

2.1.1 Robust Planning and Control

Robust controllers are defined by their ability to control a plan in the presence of bounded error to their input. At the most basic level these errors appear as initial state and control signal perturbations, but work in robust model predictive control (MPC) also attempts account for error in the system model. These controllers can be characterized as either adaptive control policies that update control gains online to provide stability in a new environment (Bemporad & Morari, 2007). Such systems are tested and/or designed through stress-testing by perturbing a controllers assumptions of a robot's state, control signal, environment geometry, or contact data and finding a controller that works for the majority of cases under those assumptions. Robust control has been used for improving the reliability of or reducing uncertainty of locomoting systems (Wang et al., 2009; Mombaur et al., 2005; Saglam & Byl, 2014; Burden et al., 2015), effecting grasping behaviors (Kim et al., 2013; Mahler et al., 2015; Weisz & Allen, 2012; Zheng & Qian, 2005), and planning trajectories that reduce system uncertainty (Johnson et al., 2016). Validating such robust controllers through stress-testing, commonly known as *falsification*, seeks to find counter examples to the robustness claims of a controller (Branicky et al., 2006; Abbas et al., 2013; Esposito et al., 2005).

While robustness can be the objective of an optimization approach this dissertation and others (Schwarm & Nikolaou, 1999) seeks to explicitly constrain the sign of a robot's control system for an expected probability of failure. These methods either: (1) reject plans that fall below some threshold on robustness (e.g., failure rate, proximity to failing state) or (2) use the constraints in a linear or quadratic programming framework to locate a feasible, but likely sub-optimal plan for the robotic system subject to uncertainty (Blackmore, 2006).

Sometimes the growth of state uncertainty in a controlled system is not fixed but varies, possibly depending on factors in the environment. Patil et al. (2014) and Platt Jr et al. (2010) account for uncertainty during planning by correlating movement through a low information environment with high uncertainty in the outcome of the plan in an optimal control framework. Planning in the presence of known, heterogeneous regions of rich information (high certainty) in the environment can bias path planning toward a solution with more information, even if the path is qualitatively

more difficult or longer. Despite starting with high uncertainty, such plans can have a high level of accuracy in their output outcome due to the choice of information gaining steps, or uncertainty-reducing system evolution (Lohmiller & Slotine, 1998; Johnson et al., 2016).

2.1.2 Monte Carlo method and particle approaches The sampling-based approach to virtual robot testing presented in this dissertation contains similar elements to Monte Carlo method and particle-based approaches for state estimation of nonsmooth systems (Duff et al., 2011; Zhang et al., 2013; Koval et al., 2013; Li et al., 2015a,b). The extent of the similarity is that both use stochasticity in addition to probability distributions over state to generate time series datasets (*traces*) for each perturbation to dynamical system (*particle*) parameters. These works expose the interaction between dynamics and rigid contact mechanics, as developed in theory of linear complementarity systems (Shen & Pang, 2005).

Sampling-based approaches (Monte Carlo methods) that randomly perturb the state of particles (representing a controlled system with added uncertainty) has also demonstrated as an effective means of considering state uncertainty when planning; rapidly-exploring random tree (RRT) searches (Melchior & Simmons, 2007; LaValle & Kuffner, Jr., 2001), as well as trajectory optimization (Kalakrishnan et al., 2011b) use sampling-based approaches for path planning through high dimensional spaces with uncertainty in the process evolution. These approaches take advantage of search properties of randomized sampling to explore similar plans to negotiate complex or cluttered environments. Alternatives to Monte Carlo methods for uncertainty analysis include: Stochastic collocation (Xiu & Hesthaven, 2005) which relies on a smooth, simplified representation of the evolution of the evaluated system over time and a metric to appraising a generated plan; Moment methods (Walters et al., 2002) that correlate the variation of some output of interest with respect to system parameters.

2.1.3 Simulation-based planning Twigg & James (2007) uses *visual plausibility*, qualitatively undetectable perturbations to collision parameters, to generate a set of possible "worlds". Their idea is essentially the inverse of the sampling-based virtual testing approach presented in this dissertation: where this dissertation focuses on using various perturbations to a simulation to char-

acterize robotic behavior and identify possible divergences, they perturb simulations to try to find plausible, but low probability events.

Randomness in the outcome of a physical experiment and simulation might occur as a result of inertial and geometric variations in the physical robot, external force perturbations, or state estimate error combined with unexpected contact events that drive physical behavior away determinism. This dissertation has a similar focus: generating *physically plausible* (instead of *visually plausible*, Barzel et al. 1996) variations on the robotic system to determine the uncertainty in the system as well as detect the non-smooth, events that lead to divergent behavior (e.g. unintended contact).

Zickler & Veloso suggested exploring the space of control policies through high level primitives, essentially finding a best series of actions to achieve a stated goal when controlling in the presence of uncontrolled or antagonistic agents in the environment (2009). Similarly, Mordatch et al. proposed a planner that generates a trajectory planner using simulation-in-the-loop optimization for animating manipulation and locomotion tasks (2012; 2013). Posa et al. also proposed a contact-invariant approach for optimizing a trajectory with intermittent contact for complex manipulation and locomotion tasks (2014).

2.2 Automated robot controller and mechanism design

2.2.1 Evolutionary robotics Evolutionary robotics seeks to implement a morphology exploration strategy of iteratively modifying a robot's model by optimizing over a fitness function (Nelson et al., 2009). Constraints on robot model limits can be incorporated into the problem in two ways; (1) by only generating feasible offspring so these limits are never violated; or (2) penalizing infeasible offspring proportionally to their infeasibility. The latter approach allows for a larger search space, possibly avoiding bug-trap like topologies of the constraint space or by becoming trapped in local minima. Some work in evolutionary design simultaneously improves both the morphology and control policy of a robot. These automated robot modification systems use geometric primitives to develop robots that accomplish simple tasks such as trotting or pushing an

object (Bongard, 2014; Sims, 1994).

2.2.2 Morphological computation The virtual creatures generated through a genetic algorithmic approach (see Auerbach & Bongard 2012) shed light on a deeper insight that morphological design may have just as much sway on certain aspects of control as modifications to the control system itself. Changes to a robot's morphology affect its physical behavior, but do not accrue additional computational cost, they are computationally relevant for control and stability of the locomoting system (Sims, 1994; Pfeifer & Iida, 2009).

2.2.3 Situated robotics & embodied cognition The interaction of a robot's morphology with its environment can have such a great impact on robot behavior, that these interactions must be anticipated when developing a robotic system (Cangelosi et al., 2015; Dorigo & Colombetti, 1994). This importance of working with *situated* systems is emphasized by work with *embodied cognition* which has sought to understand the interaction of cognition with the sensory information from a physical environment. The necessity of considering situated performance becomes especially important in the context of locomoting systems that continually make and break contact with their environments; their behavior is heavily determined by how those interactions affect the system dynamics (Cheney et al., 2016).

***2.2.4 Validity of programmed behavior* in situ** The *T-Resilience* (Transferability based resilience) algorithm (Koos et al., 2013a) attempts to discover compensatory behavior in unanticipated situations (i.e., alternative strategies or maneuvers to achieve the same task—such as hopping on one leg to locomote if the other leg is injured) toward maintaining performance between virtual and physical (*in situ*) tests. Their approach searches a "transferability map" which attempts to characterize the relation between the robot's programming and the physically expressed characteristics and behavior of a robot (Koos et al., 2013b). The virtual testing an robot modification approach presented in this dissertation effectively seeks to improve transferability by searching for a control policy or hardware configuration that has a shallow slope on the transferability map. This work aims to improve transferability by avoiding sensitive regions of a control policy with respect

to model uncertainty. The virtual testing approach presented in Chapter 3 discovers these sensitive areas in the local parameter space through a pseudorandom Monte Carlo method search. Chapter 5 details how updates the robot's kinematic, geometric and inertial parameters can steer the robot's design away from these sensitive areas in the local parameter space.

2.3 Limbed robots: Nonholonomic control with contact

The work in this thesis focuses specifically on robots that physically interact with their environment via contact (i.e., manipulation and locomotion). Contact is a governing factor for the movement of legged robots about their environment and for the manner in which robot hands pick up, move, operate, and otherwise manipulate objects in their environment. All of the experiments in this work, whether operating robots *in situ* or *in sim*, exhibit hard-to-predict making and breaking of multiple contacts. The planning and control systems are thus designed to produce a motion plan that is robust to various contact configurations.

2.3.1 Locomotion This thesis includes the implementation of a reactive operational space control strategy for locomoting systems that allows the robot to decouple the configuration of its morphology from its end effector and base configurations (Barasuol et al., 2013) assuming all operational space goals are kinematically reachable. A similar locomotion controller is presented in PACER (Zapolsky, 2015), which borrows lessons in foot placement and simple trotting behavior from Hengst et al. (2002), and uses similar parameterizations to other gait planning systems for quadrupedal robots (Coros et al., 2011; Kalakrishnan et al., 2011a).

Though this transition to operational space may lead to computationally difficult to handle redundancies in configuration space, these control redundancies can be dealt with through careful use of inverse kinematics (Peters et al., 2008; Satzinger et al., 2014). The quadrupedal robots I focus on throughout this thesis do not have redundant actuation, but these considerations are important for more complex walking robots such as bipeds.

2.4 Stability analysis and control of nonsmooth systems

A number of researchers have studied stability analysis and control of nonsmooth mechanical systems (Brogliato, 1996; Tomlin et al., 2000, 2003; Prajna & Rantzer, 2007; Prajna et al., 2007; Leine & van de Wouw, 2008a,b; Papachristodoulou & Prajna, 2009; Posa et al., 2015), for which hybrid dynamical systems have been a common formal model. These systems have been applied toward the study of walking machines and robots, which this dissertation also uses as an illustrative application.

2.4.1 Bifurcations in dynamical systems Nonlinear dynamic systems occasionally exhibit diverging behavior between two similar system parameterizations. Systems parameterized at these points of divergence (bifurcations) will exhibit large changes in behavior resulting from small deviations in state. If the parameters of such systems are subject to uncertainty, such as error resulting from manufacturing tolerances or noisy sensory information, the behavior of these systems becomes hard to predict. Locating such bifurcations in large parameter spaces and discovering points in the parameter space that are maximally distant from all such bifurcations is one method of limiting the appearance of divergent behavior in these systems. Smith et al. compute the "near-maximally sized" hyper-rectangle that is centered at a fixed parameter-state point whose elements are guaranteed to exclude all bifurcation points (2014). Their approach is based upon a branch and bound algorithm to discover the outer bounds of the bifurcating regions of a parameter space.

Extending the detection of bifurcations in nonlinear dynamic systems to algebraically constrained dynamical systems offers a method of discovering points of divergent behavior in nonsmooth systems. A power system (i.e., the control system for an electrical power grid) is assessed in (Venkatasubramanian et al., 1993). Similarly to the nonlinear dynamic systems, when the system parameters are changed, a topological shift in the behavior of the system is triggered. For the operation of controlled physical systems which are controlled about a stable equilibrium point, such systems can lose dynamic stability when they encounter a bifurcation in their parameter space.

2.5 The rigid body

The work in this thesis considers robot dynamics that are well modeled by rigid bodies and rigid or nearly rigid contact but is not generally predicated on these assumptions. Deformable body simulations, for example might provide a more representative albeit far slower model of a particular robotic system.

2.5.1 Rigid body dynamics The multi rigid body dynamics equation governing the dynamics of a robot undergoing contact can be written in its generalized form as:

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} = \mathbf{P}^{\mathsf{T}}\boldsymbol{\tau} + \boldsymbol{f}_{\text{ext}} \tag{1}$$

$\boldsymbol{f}_{\text{ext}} \in \mathbb{R}^m$ is a vector of "external", non-actuated based, forces on the m degree-of-freedom multi-body, like gravity and Coriolis forces. $\boldsymbol{\tau} \in \mathbb{R}^{nq}$ is a vector of actuator torques (where nq is the number of actuated degrees of freedom). $\mathbf{H}(\boldsymbol{q}) \in \mathbb{R}^{m \times m}$ is the generalized inertia matrix of the robotic system. $\mathbf{P} \in \mathbb{R}^{nq \times m}$ is a binary selection matrix, mapping the nq controlled degrees of freedom to the m generalized degrees of freedom of the robotic system. If all of the degrees-of-freedom of the system are actuated $\mathbf{P}$ will be an identity matrix. For, *e.g.*, legged robots, some variables in the system will correspond to unactuated, "floating base" degrees-of-freedom (DoF); the corresponding columns of the binary matrix $\mathbf{P} \in \mathbb{R}^{(m-6) \times m}$ will be zero vectors, while every other column will possess a single "1".

2.5.2 Non-smooth mechanical systems In addition to the challenges of analyzing nonlinear dynamics stability (of multi-rigid body systems), the problem discussed in this thesis requires consideration of *nonsmooth mechanical systems* (Brogliato, 1996), for which velocities can change discontinuously due to impacts and even non-impacting contact with Coulomb friction (Stewart, 2000a).

Multi body dynamics with rigid contact and Coulomb friction—which captures important stick-slip transitions—can be modeled as piecewise differential algebraic equations (DAE). When taking

into account unilateral constraint (e.g., contact, joint position limits), Equation 1 extends to:

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} = \mathbf{P}^{\mathsf{T}}\boldsymbol{\tau} + \boldsymbol{f}_{\text{ext}} + \mathbf{J}^{\mathsf{T}}\boldsymbol{f}_C \tag{2}$$

$$\Phi(\boldsymbol{q}) \geq \mathbf{0} \perp \boldsymbol{f} \geq \mathbf{0} \tag{3}$$

$$\boldsymbol{f}_C = [\boldsymbol{f}_N^{\mathsf{T}} \quad \boldsymbol{f}_S^{\mathsf{T}} \quad \boldsymbol{f}_T^{\mathsf{T}}]^{\mathsf{T}} \text{ (contact forces)} \tag{4}$$

$$\mu^2 \boldsymbol{f}_N^2 = \boldsymbol{f}_S^2 + \boldsymbol{f}_T^2 \text{ (Coulomb friction)} \tag{5}$$

$\boldsymbol{f}_C \in \mathbb{R}^{3nc}$ is a vector of contact forces which are decomposed into normal "$_N$", and tangential "$_S$", "$_T$" directions (where nc is the number of active contacts). $\mathbf{J} \in \mathbb{R}^{3nc \times m}$ is a Jacobian matrix, where $\mathbf{J}$ maps from generalized velocities to contact velocities and $\mathbf{J}^{\mathsf{T}}$ maps from contact forces to generalized forces acting on the robotic system.

Contact forces are unilateral and are subject to the complimentarily constraint in Equation 3, where $\Phi(\boldsymbol{q})$ is the gap function for the minimum distance between two geometries; $\Phi(\boldsymbol{q}) = 0$ indicates that the two geometries are in contact (the constraint is active and contact forces can be non-zero), and $\Phi(\boldsymbol{q}) > 0$ indicates that the geometries are not in contact. Some constraints are always active over a time interval, like bilateral joint constraints. Other constraints are only active if certain conditions are met; e.g., a contact constraint between two bodies would only be active when the bodies are in contact at that point and they would otherwise (i.e., without the constraint in place) interpenetrate at that point. Other algebraic constraints acting on a robot system may include motor torque and velocity limits, which constrain how much torque can be applied to the system through actuation. This latter kind of problems can be formalized in the hybrid dynamical systems context as a differential complementarity problem (Pang & Stewart, 2008):

2.5.3 Simulating multi-rigid bodies The multi-rigid body dynamics are assumed to be integrated over interval Δt. Using a first-order discretization of the differential algebraic equations modeling

the system a half-explicit, index 3 DAE (Hairer & Wanner, 1996) is defined:

$$\boldsymbol{q}_{i+1} \leftarrow \boldsymbol{q}_i + \Delta t \dot{\mathfrak{q}}_i \tag{6}$$

$$\dot{\mathfrak{q}}_{i+1} \leftarrow \dot{\mathfrak{q}}_i + \Delta t \boldsymbol{f}(\boldsymbol{q}(t), \dot{\mathfrak{q}}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}) \tag{7}$$

$$\boldsymbol{\Phi}(\boldsymbol{q}_{i+1}, \dot{\mathfrak{q}}_{i+1}) \geq \mathbf{0} \quad \perp \quad \boldsymbol{\lambda} \geq \mathbf{0} \tag{8}$$

where $\boldsymbol{f}(.)$ yields the time derivative of generalized velocities as a function of generalized coordinates, generalized velocities, actuator forces, and some constraint forces. Spatial vector $\dot{\mathfrak{q}}$ is defined in the following section (§2.5.4). Vector $\boldsymbol{\lambda}$ acts as a Lagrange Multiplier (i.e., it is positive if the constraint is active and zero otherwise) and $\perp$ indicates the complementarity condition $\Phi_i(.)\lambda_i = 0$. From (1) the distinctions between generalized coordinates and generalized velocities and (2) the algebraic constraints $\boldsymbol{\Phi}(.)$, it can only be expected that $\boldsymbol{q}(t + \Delta t) \approx \boldsymbol{q}(t + \Delta t)'$ and $\dot{\mathfrak{q}}(t + \Delta t) \approx \dot{\mathfrak{q}}(t + \Delta t)'$: strict equality is not ensured, though $\Delta t \ll 1$ and a first-order approximation of the next velocity when using inverse dynamics control bounds the divergence from a commanded trajectory when using this approach (Chapter 7).

2.5.4 Spatial and generalized velocities Note that this dissertation assumes that spatial velocities ($\dot{\mathfrak{x}} \in \mathbb{R}^6$) and generalized velocities ($\dot{\mathfrak{q}} \in \mathbb{R}^{nv}$) are not equivalent to the time derivatives of rigid body coordinates ($\dot{\boldsymbol{x}} \in \mathbb{R}^7$) and generalized coordinates ($\dot{\boldsymbol{q}} \in \mathbb{R}^{nq}$), respectively, as the focus of this dissertation is locomoting (underactuated) robots and there exists no physically meaningful, minimal representation of orientation in 3D (Kane & Levinson, 1985). The 3D orientation of the robot's "floating base" is represented using unit quaternions, the rotational velocity using a 3D angular velocity vector, and convert between the two using linear operations (Nikravesh, 1988):

$$\dot{\boldsymbol{x}} = \mathbf{N}^* \dot{\mathfrak{x}} \tag{9}$$

$$\dot{\mathfrak{x}} = \mathbf{N}^{*\dagger} \dot{\boldsymbol{x}} \tag{10}$$

and

$$\dot{\boldsymbol{q}} = \mathbf{E}^{*}\dot{\mathfrak{q}} \tag{11}$$

$$\dot{\mathfrak{q}} = \mathbf{E}^{*^{\dagger}}\dot{\boldsymbol{q}} \tag{12}$$

where $\mathbf{N}$ and $\mathbf{E}$ are left-invertible, block diagonal matrices dependent upon $\boldsymbol{x}$ and $\boldsymbol{q}$, respectively, and the † operator indicates pseudo-inversion (for which the left-invertibility implies that there will be no residual error).

3 Particle traces

Planning algorithms and control approaches often rely on the fidelity of computational models to the physical properties of a robot (i.e., inertia, kinematics, surface friction, link geometry, etc.) and a reasonably accurate estimate of the robot's state in the world (i.e., position, orientation, velocity). However, the physical properties and state are not generally known to high accuracy. Fabrication imprecision, damage to parts, control lag, noisy sensors, communication delays, and sensor drift are only a few sources of error that can differentiate the behavior of simulated robots from reality. This chapter describes an approach that uses simulation to detect possible such behavioral divergences on real robots. This approach, and others like it, can be applied to validation of robot behaviors, mechanism design, and even online planning, as I will show in Chapters 4 and 6.

This chapter deconstructs a Monte Carlo method approach to simulation-based robotic testing—henceforth referred to as the "particle traces" approach—into several distinct parts: Section 3.1 overviews where errors in state estimation, control fidelity, sensor noise, and external force perturbations contribute to uncertainty in a robotic system and how these factors might be represented in simulation with a particle-driven approach; Section 3.2 discusses methods of randomly sampling robot model parameters when generating particles; Section 3.3 defines a control policy in the context of this work; Section 3.4 defines the representation of a task and its requirements for success; finally, Section 3.5 describes the particle tracing algorithm that generates each particle then simulates each particle trace for an interval of time while detecting divergent behavior. The particle traces approach description is followed by discussion of the computational complexity of Algorithm 1 that implements the approach (§3.5.1) and then a description of how particle trace telemetry data may be processed (§3.6).

3.1 Control and simulation of robotic systems subject to uncertainty

This section presents an overview of how uncertainty, control infidelity, communication delay, and measurement and fabrication inaccuracy in a robotic system are considered in this dissertation; the section introduces the sampling-based "particle trace" approach. Controlling a robotic sys-

tem *in situ*—with imperfect modeling assumptions, sensor models, motor models, etc.—inevitably leads to divergence between a robot's intended and current state. The behavior of the robot described by a planner and the actual behavior of the physical system diverge as these factors compound and introduce uncertainty into the robotic system. Figure 1 illustrates how these factors contribute to uncertainty in the typical sense-plan-act paradigm of robotic planning, estimation, and control.

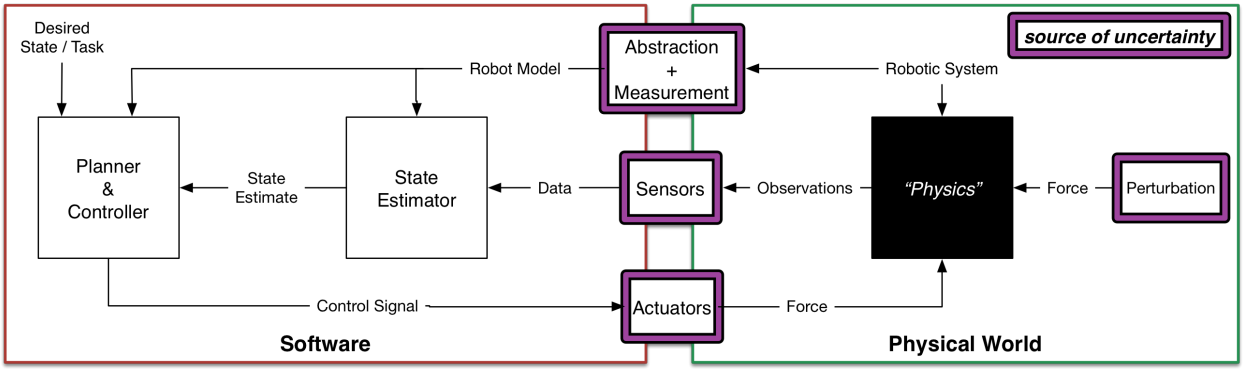


Figure 1: ***Sources of uncertainty*** **in situ.** ***The fidelity of a control system's expected behavior to its performance*** **in situ** ***will depend on how much each "source of uncertainty" perturbs the information passing through it.***

Figure 2 illustrates where uncertainty is added to a simulated robotic system; this approach is meant to emulate the sources of uncertainty in a situated system (as in Figure 1). Beyond the initially evident discrepancies between these two diagrams, temporal disagreements (e.g., the duration of time required to integrate the equations of motion in a physical simulation, communication delays in controllers and sensors) further separate the virtual and *in situ* robotic control paradigms from one another as they contribute hard-to-model uncertainty into the robotic system (Taylor et al., 2014). The degree of correlation between the behavior of robots simulated over a timespan of seconds or minutes and those robots' physically situated counterparts depends on many factors. A somewhat flexible robot may evince little of the behavior of its virtual counterpart simulated using multi-rigid-body dynamics, for example. Validation of multi-rigid body dynamics simulations (and other kinds of robot simulations) is a new effort and has been explored by Taylor & Drumwright (2016) and Yu et al. (2016). The following sections describe the theory and formulation behind the particle traces approach.

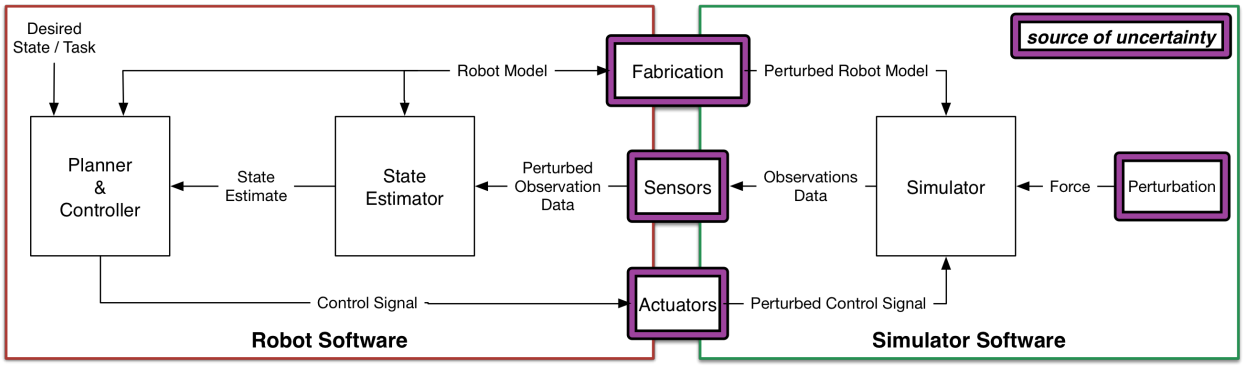


Figure 2: ***Sources of uncertainty in simulation. The sampling-based approach described in this dissertation introduces the box labeled "Fabrication". Fabrication, in this context, performs an inverse function to "Abstraction + Measurement" in Figure 1. In this case, there is an ideal design of the robot on which the software operates, but creating this design in reality is difficult; the physical or simulated system will differ in behavior slightly from the ideal system.***

3.2 Sampling-based approach

The evolution and interactions between the mentioned elements of a robot control system are complex; quickly simulating the behavior of robots and other objects or mechanisms inevitably requires us to rely on computationally efficient abstractions to the physical system—such as the rigid body assumption, or Coulomb friction. Rather than introducing new parameters to these models, or defining yet another model (increasing the complexity and number of tuned parameters in the simulated system), a sampling-based approach is followed, toward exploring the complex dynamics of these robotic systems. The sampling-based approach simulates many *particles*, each of which is characterized by a perturbation to the calibrated or measured parameters of the models describing a robotic system. The fundamental unit of this approach is the particle, which describes each instance of a robot or mechanical system (i.e., environmental, kinematic, dynamic, geometric, control, sensing, and communication) with a set of parameters.

3.2.1 Generating particles The particle traces approach perturbs the physically simulated model of the robot (i.e., using parameters $\boldsymbol{p}^*$), but keeps the controller's belief about its parameters at their expected values $\boldsymbol{p}_0$; this emulates the disagreement between the virtual model and physical hardware of a robot that is expected during situated testing. The physical system or the parameters of the physically simulated system during the execution of Algorithm 1, may differ from the kine-

matic, dynamic, and geometric models utilized by the controllers.

Each particle parameter is perturbed from its initial value ($\boldsymbol{p}_0$) using a sampled perturbation *before* starting the simulation. A particle is then *traced* over time as the virtual robot follows its control policy within simulation. Sensor noise and control lag jitter[1] are additionally perturbed *on each control loop iteration*. Figure 3 illustrates this sampling process for a single particle parameter. Each particle is traced from its initial state at the current time to a user-specified time in the future $t^{\max}$. Algorithm 1 describes the process of tracing these sampled particles and processing their telemetry.

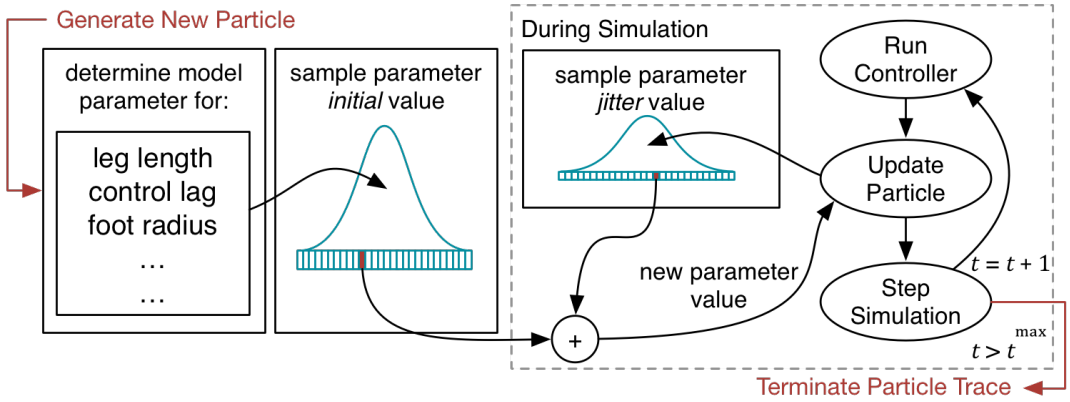


Figure 3: ***Example of the life-cycle of a particle during the execution of Algorithm 1. When a new particle is generated: (1) persistent values of parameters are determined before the first simulation update; (2) transient parameters that vary per-control loop are perturbed from a persistent mean value at each simulation update.***

3.2.2 Pseudorandom sampling of particle parameters

Each particle's parameter values are generated by pseudo-random sampling (quasi-random sampling is considered in Section 3.2.3) on each of the uncertain elements of a robotic simulation. The usage examples in Chapter 4 demonstrate what parameters are considered for different robots performing various tasks.

Measured, or desired parameter values (e.g., limb length, joint axis, surface friction) are sampled from a Gaussian distribution about the measured or intended measurement μ_{p_i} with a variance $\sigma(p_i)$ determined by the precision of the measurement or method of fabrication. Transient parameter values (e.g., control lag jitter, sensor noise) are perturbed by a zero mean Gaussian with variance

[1]Control lag jitter is a small delay that is added/subtracted from the control lag and randomly selected on each control loop iteration.

$\sigma(p_{i,\text{jitter}})$ at each observation. Persistent parameters have zero jitter.

$$p_i(0) \sim \mathcal{N}(\mu(p_i), \sigma(p_i)) \tag{13}$$

$$p_i(t) \sim p_i(0) + \mathcal{N}(0, \sigma(p_{i,\text{jitter}})) \tag{14}$$

Parameters resulting from an arbitrary decision over an indeterminate set of choices (e.g., impact model, contact model) are selected randomly with uniform probability between all n choices.

$$p_i \sim \mathcal{U}(1..n_{p_i}) \tag{15}$$

The experiments described in Chapter 4 used Gaussian and uniform distributions over wide ranges to effect a safety factor (see Section 4.2).

3.2.3 Quasi-random sampling This section comments on the effectiveness of using quasi-random sampling—a Sobol sequence (Press et al., 1992) in this case—to mimic the distributions produced by random sampling while providing deterministic space coverage; quasi-random sampling is appealing because it encourages a homogenous increase in sampling density throughout the sampled volume as the number of samples increases, guaranteeing even coverage using N samples, where N is unknown *a priori*. Pseudorandom and quasi-random sampling strategies were compared on a simple example where a jointed pendulum swings past a small, fast moving object; a very small region of pendulum initial states—nearby the specified initial state—lead to the pendulum striking the object. When sampling from a 24 dimensional state space to detect the existence of that small-volume event in the pendulum's state space with a Gaussian initial state uncertainty, both pseudorandom and quasi-random sampling strategies yielded approximately the same results: (1) the approximate likelihood of striking the object given the initial state uncertainty and (2) the minimum number of samples drawn before the event was first detected were approximately the same between sampling strategies. Considering this preliminary result, implementing an algorithm to produce a Sobol sequence on an arbitrarily large parameter set (up to 175 parame-

ters in Figure 15) could not be justified given the lackluster improvement in sampling coverage or sampling strategy effectiveness.

3.3 Control Policy

Input forces $\boldsymbol{u}(t)$ are determined by a control policy π, which is dependent on the current state of the system $\boldsymbol{q}(t)$ and $\dot{\mathbb{q}}(t)$, time t, the duration until the next controller call Δt, and particle parameters $\boldsymbol{p}$.

$$\boldsymbol{u}(t) = \pi_{\boldsymbol{p}}(\boldsymbol{q}(t), \dot{\mathbb{q}}(t), t, \Delta t) \tag{16}$$

Definition of the specific control policy is considered separately for each robot experiment. The control pipeline used throughout this thesis is described in Chapter 9. Some functions are evaluated with respect to a robot's kinematic or dynamic model with parameters $\boldsymbol{p}$ (defined by the particle); the subscript "$_{\boldsymbol{p}}$" indicates with respect to which particle parameters a function or unit of data is defined .

3.4 Tasks & Task Requirements

The particle traces approach aims to predict the range of a robot's behavior on a task by perturbing the parameter values measured from the expected robotic system. Within this framework, failing a task "requirement" indicates failure during the performance of a task, and checking against a task "objective" indicates whether a task is successful once the system is given sufficient time to complete the task. A task objective is defined as a state or region of acceptable states for the robotic system and some objects in its environment. Task requirements indicate that the robot in a particle trace has performed as expected or has diverged from plan. Task requirements indicate that a particle trace has exhibited divergent behavior during the performance of Algorithm 1 by checking the state, contact, and dynamics telemetry data. Examples of a task retirements (e.g., robot striking an obstacle or falling down) are in Chapter 4, Figure 15.

3.5 Physically simulating particles: The "Particle Traces" approach

The robotic system (the robot and its environment) is represented as a nonsmooth mechanical system (Equation 2) modeled using a piecewise differential algebraic system. The process performed by a simulator (e.g., contact and physics calculation, integration, collision detection) is represented briefly as a SIMULATOR$(.)$ function that takes a state and actuation input and outputs the state of the system a duration of time t_f into the future, and a binary vector flags:

$$\{\boldsymbol{q}(t_f), \dot{\mathrm{q}}(t_f), \mathrm{flags}\} = \textsc{simulator}(\boldsymbol{q}(t), \dot{\mathrm{q}}(t), \boldsymbol{u}(t), t_f) \tag{17}$$

The vector flags is a list of true or false return values for various checks and tests supplied by the operator. The flowchart in Figure 4 describes how the SIMULATOR integrates a piecewise differential algebraic system forward in time.

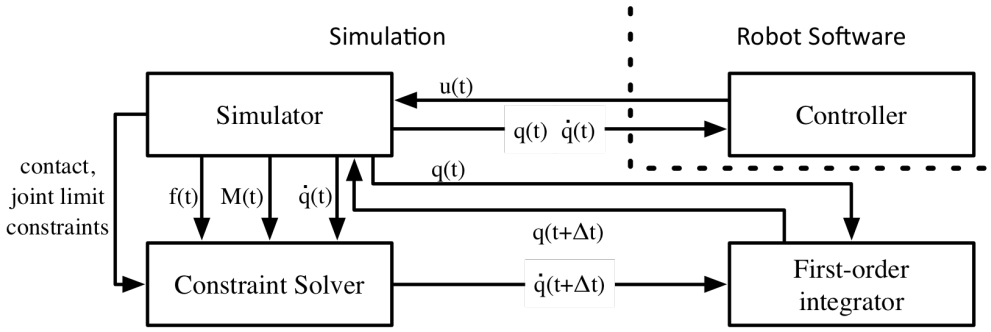


Figure 4: ***A flow chart depicting the evaluation of*** $\textsc{simulator}_p$ ***(see Equation 17). The flow-chart depicts how data flows to and from the robot's control system during integration of simulation when operating in a physically simulated environment. Vectors*** u, q, $\dot{\mathrm{q}}$, f, ***and matrix*** M ***are the actuator torques, generalized positions, generalized velocities, generalized forces, and generalized inertia, respectively.***

Algorithm 1 details an approach where success is binary: a task objective is either satisfied or not. Some robot performance is not as easily characterized so concisely, as in a robot that runs at half the intended speed without falling over may not have succeeded at its task, but the robot has not failed catastrophically either. Although careful definition of a task objective will mitigate mislabeling acceptable behavior as a failure, the particle traces approach, in its current form, better serves as a method of fault detection for discovering and mitigation of unexpected failure. The algorithm separates unknown reasons for failure from those which can be detected

Algorithm 1 $\{\mathcal{S}, \mathcal{K}, \mathcal{U}\}$ = TRACEPARTICLES$(\boldsymbol{p}_0, \mathcal{Q}, \mathit{\Pi}, N)$ Simulates N piecewise differential-algebraic systems with parameters $\boldsymbol{p}_0$ and uncertainly described by $\mathcal{Q}$ (see particle generation Section 3.2.1) for each candidate control policy π in set $\mathit{\Pi}$, while considering task requirements and the final task objective (goal state of the system). The algorithm returns particle trace telemetry data sorted into sets $\mathcal{S}$, $\mathcal{K}$, and $\mathcal{U}$, corresponding to successful, failed in progress (violated requirement), and unable to be completed (unmet objective) particle trace, respectively (see §3.6 for more information on these sets).

$\mathcal{P} \leftarrow \emptyset$
for $i \in \{1, ..., N\}$ **do** ▷ Generate samples for N particles with randomized perturbations.
  $\boldsymbol{p}^* \leftarrow$ DRAWSAMPLE$(\boldsymbol{p}_0, \mathcal{Q})$
  $\mathcal{P} \leftarrow \{\mathcal{P}, \boldsymbol{p}^*\}$
$\mathcal{S} \leftarrow \emptyset$
$\mathcal{K} \leftarrow \emptyset$
$\mathcal{U} \leftarrow \emptyset$
for each $\pi \in \mathcal{T}$ **do**
  for each $\boldsymbol{p}^* \in \mathcal{P}$ **do** ▷ Have all particles perform each task
    $\{\boldsymbol{q}_{\boldsymbol{p}^*}(t), \dot{\mathfrak{q}}_{\boldsymbol{p}^*}(t)\} \leftarrow \boldsymbol{p}^*_{\{\boldsymbol{q}_{\text{init}}, \dot{\mathfrak{q}}_{\text{init}}\}}$ ▷ Retrieve randomized initial state from particle parameters
    $t \leftarrow 0$
    $h \leftarrow$ `step_size`, where `step_size` $\ll 1$
    $\mathcal{X} \leftarrow \emptyset$ ▷ Particle telemetry "trace"
    while $t < t^{\max}$ **do** ▷ Simulate until task completion
      $\{\boldsymbol{u}_{\boldsymbol{p}_0}(t)\} \leftarrow \pi_{\boldsymbol{p}_0}(\boldsymbol{q}_{\boldsymbol{p}^*}(t), \dot{\mathfrak{q}}_{\boldsymbol{p}^*}(t), t)$ ▷ Get input forces from policy
      $\{\boldsymbol{q}_{\boldsymbol{p}^*}(t + \Delta t), \dot{\mathfrak{q}}_{\boldsymbol{p}^*}(t + \Delta t), \text{flags}\} \leftarrow \text{SIMULATOR}_{\boldsymbol{p}^*}(\boldsymbol{q}_{\boldsymbol{p}^*}(t), \dot{\mathfrak{q}}_{\boldsymbol{p}^*}(t), \boldsymbol{u}_{\boldsymbol{p}_0}(t), \Delta t)$ ▷ Integrate physical simulation
      $\mathcal{X} \leftarrow \{\mathcal{X}, \{\boldsymbol{q}_{\boldsymbol{p}^*}(t), \dot{\mathfrak{q}}_{\boldsymbol{p}^*}(t), \boldsymbol{u}_{\boldsymbol{p}_0}(t)\}\}$
      if $\exists_i$ s.t. $\text{flags}_{\text{requirement}_i} = 0$ **then** ▷ Check whether task requirements are satisfied
        $\mathcal{K} \leftarrow \{\mathcal{K}, \mathcal{X}\}$ ▷ Violated task requirement (known failure)
      $t \leftarrow t + \Delta t$
    if $\exists_i$ s.t. $\text{flags}_{\text{objective}_i} = 0$ **then** ▷ Check whether task objectives are satisfied
      $\mathcal{U} \leftarrow \{\mathcal{U}, \mathcal{X}\}$ ▷ Task is incomplete (unknown failure)
    else ▷ Final state satisfies task objective (success)
      $\mathcal{S} \leftarrow \{\mathcal{S}, \mathcal{X}\}$
return $\{\mathcal{S}, \mathcal{K}, \mathcal{U}\}$

from system telemetry, but not perfectly performing systems from those that just barely succeed at a task. The particle traces algorithm is used in Chapter 4 as a method of detecting the feasibility of a control policy or set of particle parameters when set to perform a particular task, rather than as an indicator of control policy optimality (discussed in Chapter 4). Chapter 5 explores how the output of Algorithm 1 can be used to iteratively improve a robot morphological or control system design.

3.5.1 Computational Complexity On a single task with a specified maximum time for executing that task in virtual time $t^{\max}$, tracing a thread of computation will require an amount of time equal to some scalar multiple of $t^{\max}$, corresponding to the real-time factor m of the simulation ($m > 1$ being faster than real-time). Write-conflicts can be avoided when particle trace threads of execution terminate and rejoin the main execution thread by assuming a fixed number of samples N and pre-allocating memory space for storing the telemetry data of each particle. Randomized sampling decouples individual particle trace simulation, since particle parameters and their evolution are not interdependent, they can be executed in parallel. Any number of particle traces can be generated and then executed in linear time $\mathcal{O}(\frac{N \cdot t^{\max}}{c \cdot m})$ with respect to the real time factor of simulation and the number of processor cores c available for simultaneous particle trace execution. The run time of the particle traces algorithm for one candidate control policy is then calculated as:

$$t = \frac{N \cdot t^{\max}}{c \cdot m}$$

Assuming a parallel computer with sufficiently many cores ($c \geq s$), this algorithm can run in linear time with respect to only the real time factor of the simulated system. In practice (see Chapter 4), each particle can be integrated stably in the MOBY simulator at approximately real-time speed ($m \approx 1$): each second of time in simulation (virtual time) takes about a second to compute (wall time). The conclusion from this assessment is that the particle traces algorithm can be run at real time for a very large number of samples, given a known task and sufficient computational resources—such as is becoming available with internet cloud-computing services.

3.6 Processing particle trace telemetry

Simulations are capable of generating huge quantities of data (for example, physical simulation of a locomoting quadruped for one second of virtual time produced 8.5 Megabytes of logging data in the MOBY simulator). If in the particle traces approach hundreds or thousands of particle traces are simulated over a large interval of virtual time, the abundance of telemetry data offered by the particle traces approach (Gigabytes of data) would a burden to sort through manually; to address this preponderance of information, the TRACEPARTICLES algorithm categorizes particle traces based on which task constraints they violate during execution, counting the size of the sorted information provides a preliminary look at the virtual robot's performance when subject to uncertainty.

The output of the TRACEPARTICLES algorithm (Algorithm 1) includes three sets of particle trace data $\mathcal{S}$, $\mathcal{K}$, and $\mathcal{U}$. The sets $\mathcal{S}$, $\mathcal{K}$, and $\mathcal{U}$ are mutually exclusive, and they compose all members of $\mathcal{P}$.

$$|\mathcal{P}| = |\mathcal{S}| + |\mathcal{K}| + |\mathcal{U}|$$
$$\emptyset = \mathcal{S} \cap \mathcal{K} = \mathcal{S} \cap \mathcal{U} = \mathcal{K} \cap \mathcal{U}$$

These particle trace data sets are characterized as follows: (1) Set $\mathcal{S}$ is the group of particle traces that completed the task according to all task requirements without violating a limit function during operation. (2) Set $\mathcal{K}$ is the group of particle traces that did not complete the task because they violated a limit function during operation. The best way to characterize this set is as a collection of "known failures"; these traces violated a task requirement during operation and would not be expected to perform successfully *in situ*. (3) Set $\mathcal{U}$ is the group of particle traces that did not achieve the task objective after the full interval of operation, but did not ever indicate task requirement failure before particle trace termination at $t^{\max}$. The best way to characterize this set is as a collection of "unknown failures"; the outcome of these particle traces indicates that the desired task objective was not satisfied, but the reason for this divergent behavior was not detected; all task requirements

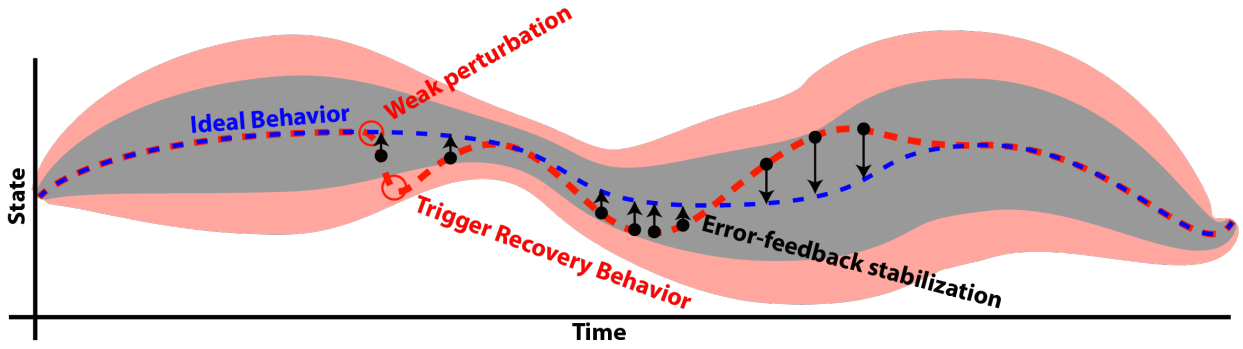
were satisfied during particle trace execution. Set $\mathcal{U}$ includes all traces that failed to satisfy the task objective but never violated any of the task requirements used to detect divergent behavior.

Chapter 5 demonstrates how members of set $\mathcal{K}$ can be used to iteratively design better mechanisms and control strategies. Little can be done to address divergent behavior observed in set $\mathcal{U}$ until the event (or lack thereof) that led the robot to fail to satisfy the task objective is discovered. What is known is that these failure come about as a result of modeled phenomena (colloquially, "known unknowns"). Unmodeled phenomenon that might result in divergent behavior *in situ* (e.g. antagonistic agents in the environment, wind), or aspects of simulation that do not match reality might be characterized as "unknown unknowns". In part, improving simulation involves incorporating these "unknown unknowns" into readily computable deterministic models, so that their effects on a robotic system can be predicted. In the context of this dissertation, the approach presented in this chapter seeks to improve the detection of the former "known unknowns" in order to detect points of failure or brittle scenarios that a control policy might encounter (in the next chapter, Chapter 4).

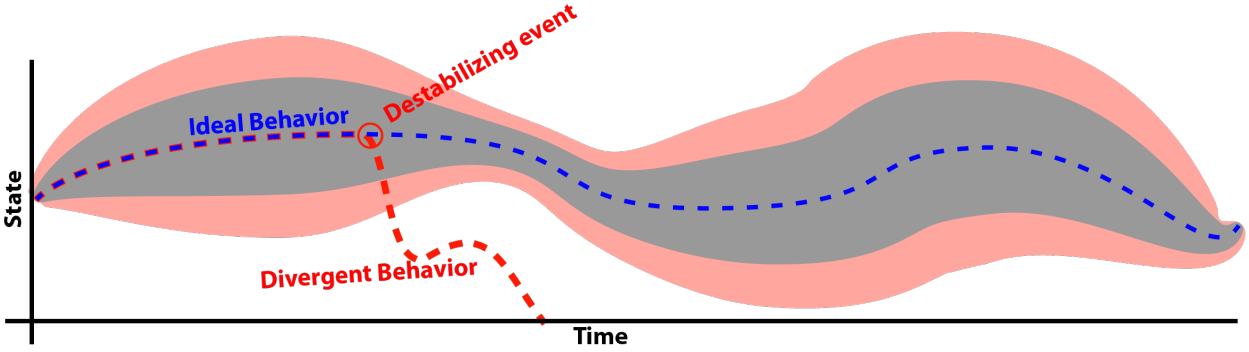
Checking the robotic system against task requirement failure at each simulation step leaves the possibility of a failure being missed resulting from a coarse discretization of time. What if the dynamics and kinematics could have detected a failure at time $\frac{t_i+t_{i+1}}{2}$? Is there a sufficiently small time interval where such events would not be missed? The solution is presented in Chapter 5; Section 5.1.2 describes an implementation of witness functions that should be designed in such a way to make missing an event unlikely. That section is also used to describe how to preemptively detect task requirement failure by tracking when a robot might exceed its limitations, possibly leading to unexpected behavior.

3.6.1 Detecting particle trace bifurcation Small modeling errors, differences in initial conditions, or both should be insignificant in the presence of effective feedback control. However, nonsmooth events can rapidly drive nonholonomic and or underactuated robots outside the region from where a closed-loop control policy can recover (see Figure 5). For a fixed-base, fully actuated manipulator executing a motion, an unexpected impact with an object may not be catastrophic to

the task performance. On the other hand, if the object is unconstrained—and underactuated—then it may be knocked out of reach of the manipulator; if the robot has a floating base (i.e., is underactuated) then it may knock itself over (e.g., a mobile manipulator pulling itself off balance, or a walking robot stubbing a toe during locomotion). All of the robot scenarios considered in this thesis are similarly underactuated, walking robots have a floating base and fixed-base manipulators attempt to pick up a floating (unconstrained) object. Therefore, a robot's behavior might be predictable if no unanticipated nonsmooth events occur (i.e., the nonsmooth events occur in all particle trace simulations or occur at similar times between particles); a robot's behavior will be harder to predict if some particle traces experience novel nonsmooth events or if nonsmooth events occurring in novel sequences.



(a) ***The trajectory of a robot stabilizing after a small perturbation. Both traces are members of $\mathcal{S}$.***

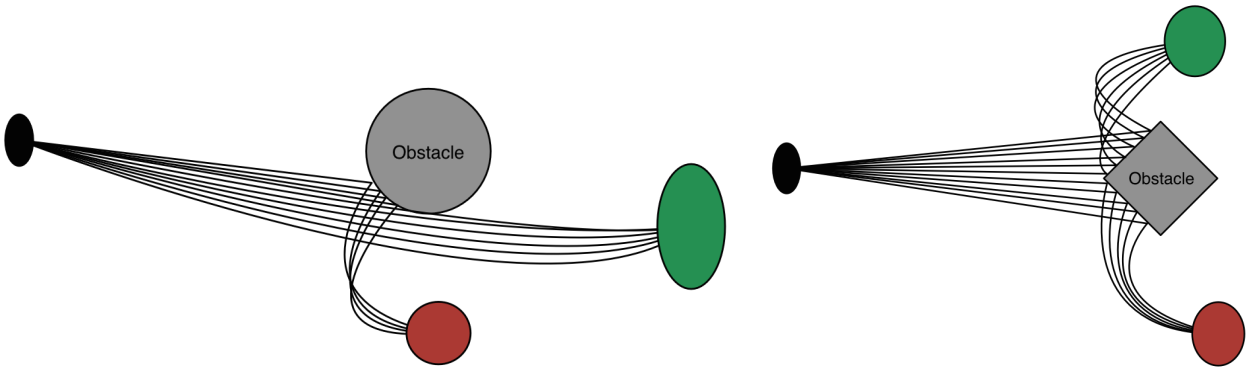


(b) ***The trajectory of a robot destabilizing after a large perturbation. The trace labeled "ideal behavior" is a member of $\mathcal{S}$, while the trace labeled "divergent behavior" would be a member of $\mathcal{K}$ if the "destabilizing event" was caught as a task requirement violation, or $\mathcal{U}$ if the destabilization remained undetected until the robot failed to achieve the task objective.***

Figure 5: ***Example of how large perturbations can push the robot away from the valid region offered by feedback control and recovery behaviors. A perturbation must be large and abrupt to destabilize a robotic system stabilized through feedback control. An impacting event in non-smooth mechanics fits these necessary qualities (e.g., unexpected contact).***

The particle traces approach can identify: (1) novel events (e.g., contact between either two links of the robot or between the robot and the environment); (2) a novel *sequence* of events; (3) novel contact between geometric features (e.g., a face-face contact between polyhedron versus face-vertex contact); and (4) disparate outcomes to the same event (i.e., indeterminacy in the evaluation of an event). *Event* is used to denote a mode switch, which can occur upon impacts and upon switching between sliding, sticking, and rolling contact. Accordingly, the presented sampling-based approach searches for both "grazing" and near-miss events (events likely to occur or not occur in only very particular conditions), collectively known in hybrid systems literature as "grazing bifurcations" (Budd, 1996). Grazing bifurcations are depicted in Figure 6.

Unexpected contact or lack of contact (Figure 6a) and indeterminate aspects (e.g., normal direction, restitution, friction) of a physical (e.g., contact, impact) model (see Figure 6b) might lead to a robot's state exhibiting divergent behavior. When a robot is operating in a region of state space near a bifurcation, the outcome of an action by the controller will generally be challenging to predict. For example, this phenomenon might occur when a slightly longer leg than is modeled scuffs a floor unexpectedly during a step or when a foot that is heavier than expected leads to tripping on a step when climbing stairs.



(a) ***Event occurrence is sensitive to error*** **(b)** ***Event outcome is sensitive to error***

Figure 6: ***The lines depict distinct particles as the systems are traced from an uncertain initial state (black ellipse, left). Each control policy aims for the system to evolve to the goal state (green ellipse, right) but—due to the differing evolutions of the system resulting from parameter differences between particles—some systems will exhibit (a) an unexpected event, or (b) an unexpected outcome of an event (when encountering the grey "obstacle") leading those systems to evolve to a failed state (red ellipse, bottom).***

State uncertainty might increases as a locomoting robot crosses a terrain feature or obstacle under open-loop control. Over a single control loop iteration $[t, t + \Delta t]$, an increase in state uncertainty could be represented as an additive noise being introduced to the process model.

$$\boldsymbol{q}_{i+1} = f(\boldsymbol{q}_i, \boldsymbol{u}_i) + \epsilon \tag{18}$$

where $\boldsymbol{q}_i$ is the state of a robotic system and $\boldsymbol{q}_{i+1}$ is the state of the robotic system after the uncertainty-increasing event affects the system over a single control loop iteration. The expected state of the robot (i.e., assuming no uncertainty) will differ from $\boldsymbol{q}_{i+1}$ by some unknown value ϵ.

Scattering Terrain features or group of features perturb the state of a locomoting robot crossing them can sometimes be manifested as white noise (Qian & Goldman, 2015). The perturbation ϵ applied to the robot's process model by a *scatterer* could be represented as unimodal Gaussian uncertainty:

$$\epsilon \sim \mathcal{N}(\mu, \sigma) \tag{19}$$

Where μ and σ are the bias and variance of the state uncertainty for each unconstrained degree of freedom of the system. A non-zero bias (μ) would indicate that there is drift in the state of robot (i.e., white noise and control input are not the only effects on the evolution of the robotic system).

Bifurcation In the spirit of Qian & Goldman I suggest a terrain geometry may be characterized as a *bifurcator*, which split (bifurcate) the expected output state of an event into two or more distributions. A bifurcating perturbation would perhaps be manifested as a bimodal Gaussian uncertainty. I conducted an experiment (to be described in §4.3.2) that assesses the ability of a simulation to help locate such bifurcations. I expected that the bifurcations detected through this method would be "grazing bifurcations", as the model parameters were perturbed around their expected values. If a robot exhibits divergent behavior after a perturbation, this means that the system was close enough to a bifurcation to make the evolution of the robotic system uncertain given a confidence in the measurements used to calibrate a robot's model.

3.7 Conclusion

The particle traces approach presented in this chapter perturbs robot modeling parameters, sensory readings, state estimates, and environmental parameters to evaluate a robot's behavior statistically over a range of conditions. Since each particle trace is completely independent, particle traces can be generated in an "embarrassingly parallel" manner. Not only is the particle traces approach versatile and simple to implement, it can be quite fast given sufficient computational resources. I believe the following questions now require much deeper investigation: (1) What dynamic scenarios can statistical ensembles of physically simulated robots reliably characterize (and where will such simulations fail to characterize behavior)?; (2) Since simulations are capable of generating huge quantities of data, how can such state space telemetry data be efficiently "mined" and how can that that be utilized to improve a robot's performance? The former question is addressed in the following chapter, where the execution of particle traces approach demonstrated in a few robot locomotion and grasping scenarios (Chapter 4). The latter question is addressed in Chapter 5 by developing a robot and control policy design suite that utilizes particle trace data to improve robot performance; validation of that approach is presented in Chapter 6. In the following chapter (Chapter 4), I will demonstrate that combining even coarse estimates of state and modeling parameters with fast multi rigid body simulation can be sufficient to detect divergent robot behavior and characterize robot performance in the real world.

4 Virtual Falsification: checking for faulty robot behavior in sim

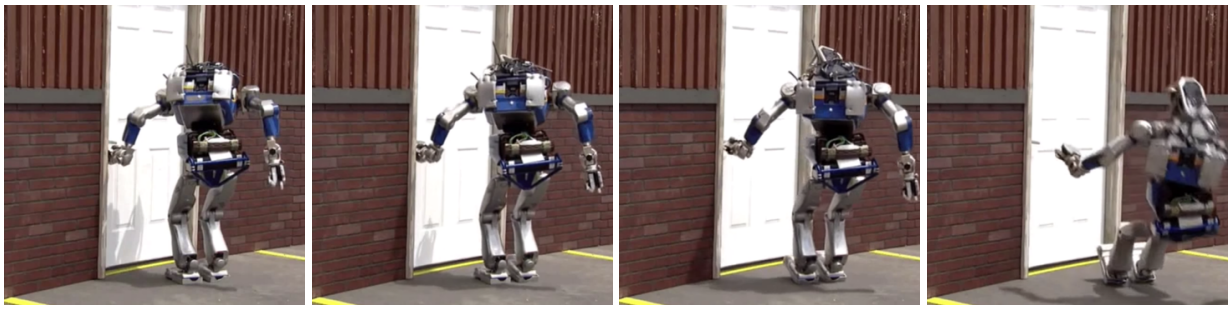
The standard approach to validating robot behavior is simulated testing followed by *in situ* testing. This approach does not inspire confidence, as simulations often fail to reflect real world behavior and *in situ* testing is tedious and slow. This problem has instigated research into formal verification methods for robotics (e.g., Johnson & Kress-Gazit 2015; Posa et al. 2015), which appears promising; intense study is currently attempting to scale these approaches to higher degree of freedom systems. This chapter explores an alternative path that is straightforward, easily implemented, and uses techniques already familiar to many roboticists to bridge the extremes of isolated physical simulation tests and full-on testing using real robotic hardware. I envision a "virtual falsification" phase during robotic software prototyping and testing where points of failure in planning and control software are identified in simulation. As the photos in Figure 7 depict, the unexpected presence or absence of contact can cause catastrophic failure.[2]

The nonsmooth mechanical systems that are considered in this chapter preclude use of existing control theoretic tools for falsification, like Branicky et al. (2006), Esposito et al. (2005), Abbas et al. (2013), and Smith et al. (2014). Virtual falsification applies multi-rigid body dynamics simulation to many perturbed versions of a robotic system (the *particle traces* approach). Virtual falsification will provide an expectation of such possible divergent behaviors that the robot might exhibit during operation (helping prevent the scenarios seen in Figure 7), *even if the simulation cannot accurately predict the exact state space evolution of the physical system.* A collection of possible robot state space evolutions may be the best that simulation can offer; even in this non-ideal case, this chapter demonstrates that virtual testing can be very informative of control policy robustness.

In this chapter, I demonstrate how the particle traces approach can be applied to various, high-dimensional, non-smooth robotics applications. Section 4.1 describes a method for processing the telemetry output of the particle traces approach toward virtual falsification. Section 4.2 describes

[2]Russ Tedrake claimed that this problem was a dominant cause of failure of the robots in DARPA's Robotics Challenge in a plenary session at Humanoids 2015.

how parameter distributions are selected for pseudorandom sampling when applying the particle traces approach to the robotics scenarios in this chapter, including: in Section 4.3 the particle traces approach is demonstrated assessing the robustness of a control policy for virtual manipulator robot performing a picking task (§4.3.1), a virtual quadrupedal robot stepping over a curb obstacle while performing a locomotion task (§4.3.2), and an *online* implementation of the approach is demonstrated on a virtual quadruped robot switching between gait strategies (§4.3.3); Section 4.4 presents a validation experiment which demonstrates that the particle traces approach can be used to detect whether a locomotion control policy will cause a robot to fall *in situ*. *Results indicate that novel, divergent behavior can be identified efficiently with even a small number of samples*.



(a) ***Turning a door handle***

(b) ***Grasping then rotating a valve***

(c) ***Stepping out of vehicle***

Figure 7: ***Robots performing tasks with anticipated contact (images captured from a video of DARPA's robotics challenge). The control strategies quickly diverge from plan without the anticipated contact, and the robots fail catastrophically. The particle traces approach can be applied to identify such brittle aspects of a plan.***

4.1 Policy scoring: success rate

In this chapter, *brittle* behavior is defined as a propensity of a robot's behavior to diverge from a desired task in the presence of small sensing, state estimation, or model calibration errors. The particle generation process (see §3.2.1) provides randomly sampled particles that exhibit these small errors; a robot's propensity to diverge from a desired task can be measured by counting the number of particle traces that exhibit an intended behavior (task objective) out of the total number of particles traces executed ($\text{score} = \frac{|\mathcal{S}|}{|\mathcal{P}|}$). A control policy that exhibits an expected behavior

in few traces is brittle where a high rate success would indicate robustness. The particle traces approach with policy scoring is probabilistic. There is no guarantee that the statistical distribution of simulated behavior will be qualitatively similar to the physically observed behavior (although such congruence has been observed in an experiment, see Section 4.4).

This "scoring" approach can help support one of two assumptions: (1) the robot will behave as expected or (2) the robot's behavior will rarely match expectation *in situ*. A representative number of particle traces $|\mathcal{P}|$ would need to be generated and executed to prove a hypothesis containing these assumptions. While "weaker" than a proof (which would require a representative sample size), the particle traces approach with a small sample size is readily applicable to the challenging problem of analyzing high dimensional systems that undergo nonsmooth behavior. Considering the large parameter space of typical robotics applications (see Figure 10 with 115 parameters and Figure 15 with 175 parameters), a statistically significant result from the particle traces approach would require many samples. Although a statistically significant sample size is large, the particle traces approach does not preclude the possibility of generating and executing an arbitrarily large number of traces in real time; the complexity and process independence of Algorithm 1 permits the parallel execution of all traces—no matter the number of samples—in real time (see §3.5.1) given significant computational resources. Nevertheless, this chapter focuses on selecting a control policy that maximizes the success rate (score) of a task with as many particles traces as can be executed while permitting rapid (but still non-realtime) use of the TRACEPARTICLES algorithm on readily accessible multi-core hardware.[3]

4.2 Enacting a safety factor for measured particle parameters

The particle traces approach generates particle parameters by pseudorandom sampling about each parameter's expected value. A parameter's expected value might be obtained by a human performing measurements on the robot offline or by the robot through automated calibration online. Perturbations to these initial measurements are produced by drawing a particle's parameter value from a truncated Gaussian distribution with mean μ equal to the measured value and variance σ

[3] 2.4 GHz Intel Core i7, MacBook Pro

set to the expected error on the measurement (see §3.2.2). The variance of each parameter is estimated conservatively; the variation in each robot parameter was determined by setting the standard deviation of the parameter's distribution to the most accurate figure on a particular measurement tool (i.e., one tick on a ruler, the kerf of the blade cutting a part to length, the resolution of a 3D printer). Enacting this safety factor accounts for measurement and fabrication error and will potentially engender a range of virtual robot behaviors similar to the behavior of multiple robots built using a single fabrication and measurement process. No attempt was made to tune the distribution parameters further from the initial estimate, as the number of parameters in even simple robotic systems would likely make such tuning infeasible. An example of the numerous parameters for a quadruped (175 parameters) is listed in Figure 15.

4.3 Illustrative & motivating examples: using the particle traces approach *in sim*

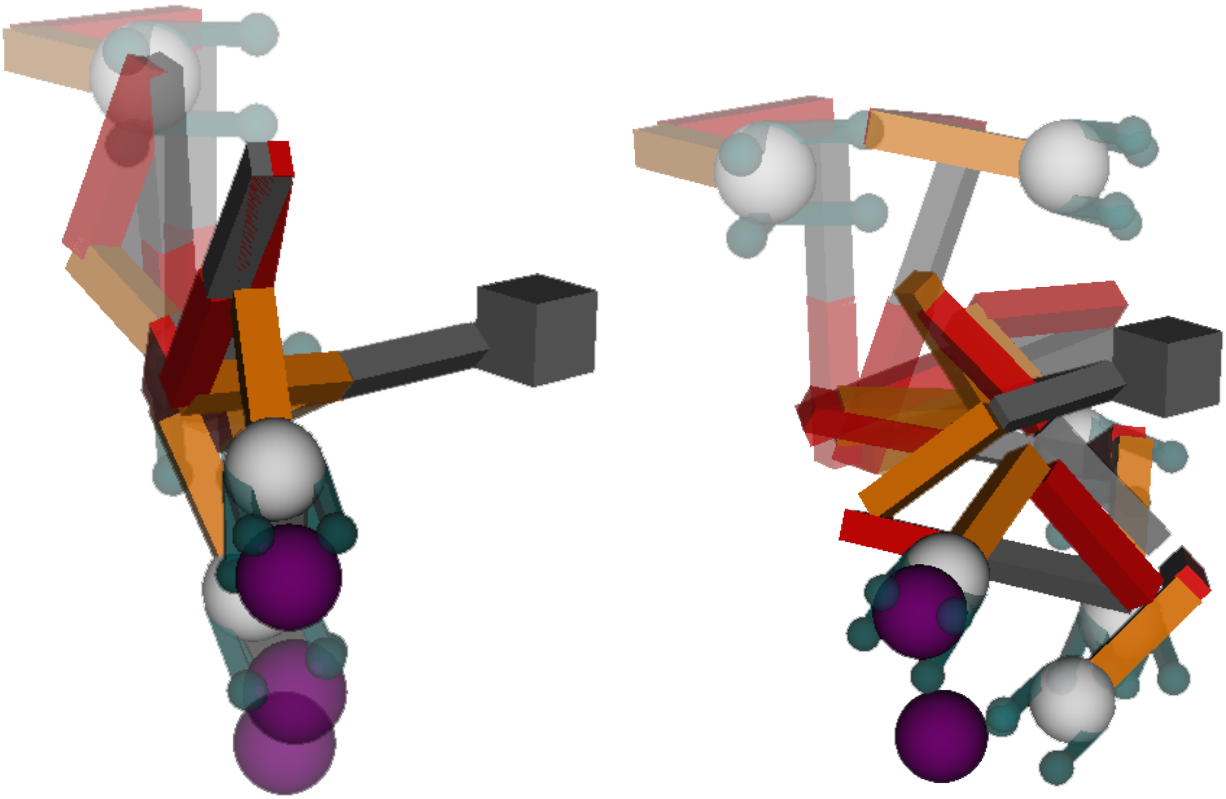
The following sections demonstrate how particle traces can be used to efficiently locate bifurcating events (§4.3.2), and assess plan robustness (§4.3.1). Walking experiments were chosen to analyze the particle traces approach because of the hybrid dynamics nature of locomotion tasks with constant making and breaking of contact and possibility of unexpected collisions; this feature of limbed robotic systems makes their simulation more complex than typical simulation used in aerospace or automotive applications. A manipulation scenario is presented because simulation-based plans for grasping have repeatedly proven to be brittle *in situ* (as depicted in Figure 7).

The particle traces approach provides confidence that a robot with numerous, imperfectly measured modeling parameters will perform successfully on high dimensional tasks. Later, in Section 4.4, I validate that the particle traces approach can be used to determine a task's robustness on a controlled robot *in situ*. The following experiments use the multi-body dynamics simulator `Moby`, which has been shown to produce behavior consistent with real robots (Aukes et al., 2014), because it uses continuous collision detection (Mirtich, 1996), allowing it to locate contact events precisely (see Zapolsky & Drumwright 2015).

4.3.1 *Manipulator: comparing policies for picking-up a ball*

An eleven-jointed fixed-base manipulator robot was simulated performing a picking task (i.e., reaching, grasping, and lifting) on a ball within its reach. Two distinct policies were followed to perform the task; the particle traces algorithm was then run to assess each policy, each of which used forty particles. Using forty particles allowed all particle traces for a policy to be executed in approximately ten seconds. Referring back to the expected computation time for Algorithm 1 (see Equation 3.5.1): given an eight virtual core machine $c = 8$, the simulation's real time factor for this robot $m = 1$, and the desired run time per policy $t = 10$ seconds, and given that a policy takes about two seconds of wall-time to execute ($t^{\max} = 2$), $N = 40$ particle traces can be run before exceeding the expected runtime per policy.

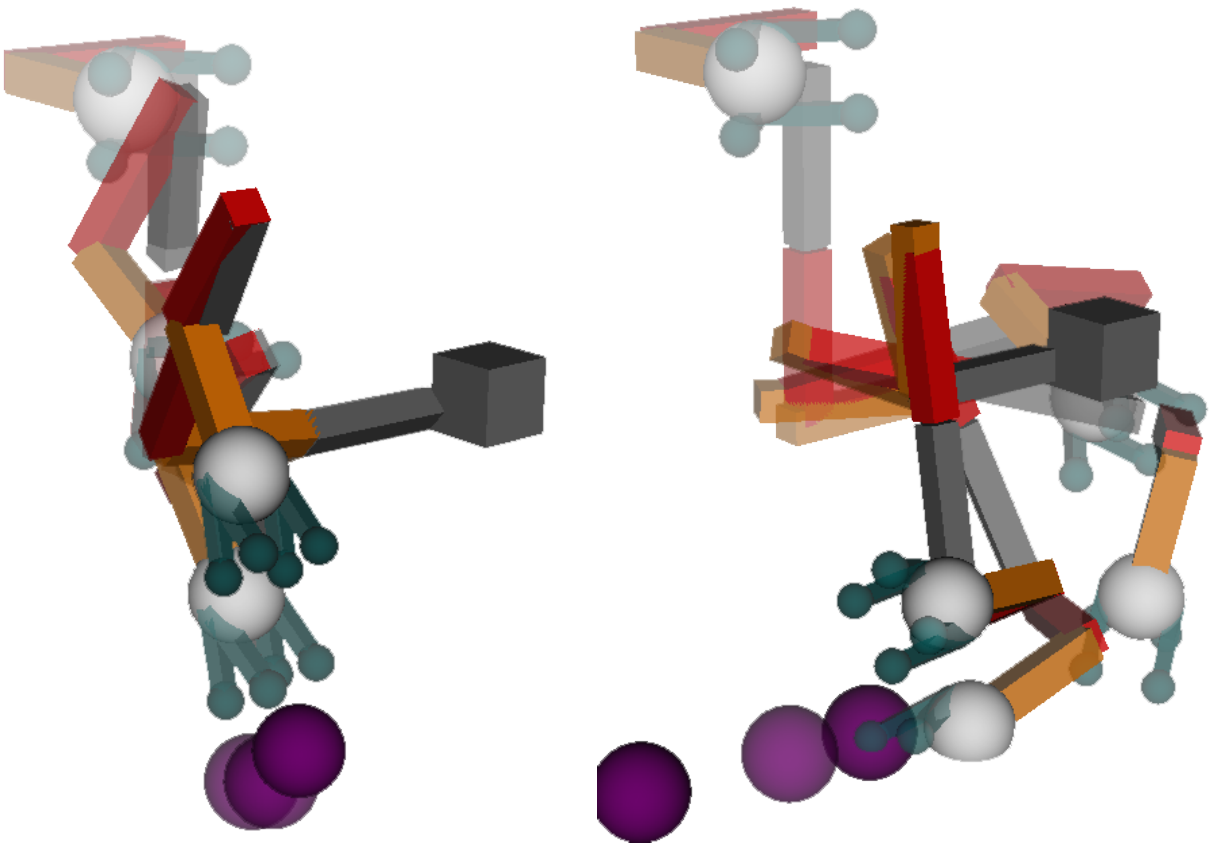
Compared Policies Two distinct policies were followed to achieve the picking task: (1) Policy A directs the gripper to move in a straight line from the gripper's initial position, toward the expected position of the ball; (2) Policy B moves the gripper to a point horizontally aligned with the ball, then approaches to grasp the ball from its side; the policy is similar to the "orthogonal approach angle" strategy from Rombokas et al. (2012). Both policies result in the unperturbed robot (parameterized as $\boldsymbol{p}_0$) successfully picking up the ball. These policies might correspond to a brittle plan generated using existing techniques. Figure 8 demonstrates the successful performance of these two policies operating in a simulation without uncertainty.



(a) ***Policy A*** **(b)** ***Policy B***

Figure 8: ***Policy A and B attempting to grasp the ball; successful performance of the policy has the gripper maintain hold on the ball.***

Task Objective & Requirements A trace was marked as successful (i.e., the robot in the trace satisfied the task objective) if the two-second policy ($t^{\max} = 2$) was completed with the gripper grasping the ball. Figure 9 demonstrates the unsuccessful performance of these two policies under imperfect conditions. Additionally, a task requirement was included for avoiding self-collision, in the interest of constraining for control policy safety and realism; neither policy ever violated this requirement. See Figure 10 for a list of the task objectives and requirements for this scenario.



(a) *Policy A, Objective Failure* (b) *Policy B, Objective Failure*

Figure 9: ***Policy A and B attempting to grasp the ball; failures drop the ball or push it away.***

Particle kinematics and Parameters The virtual manipulator robot has a fixed base and eleven dynamically simulated links connected by seven revolute joints and four revolute finger joints (one at the base of each finger). There are a total of 115 parameters determining geometric, kinematic and dynamic properties of the robot and its environment; these parameters are listed in Figure 10.

Picking task (reaching, grasping, and lifting)

Control Policy: *operational space gripper trajectory*

Policy A (Direct)
Policy B (Indirect)

Task Description

Objective:
Ball is in contact with gripper at time $t^{\max}$
Requirement:
Non-adjacent robot links do not contact one-another (avoid self-collision)

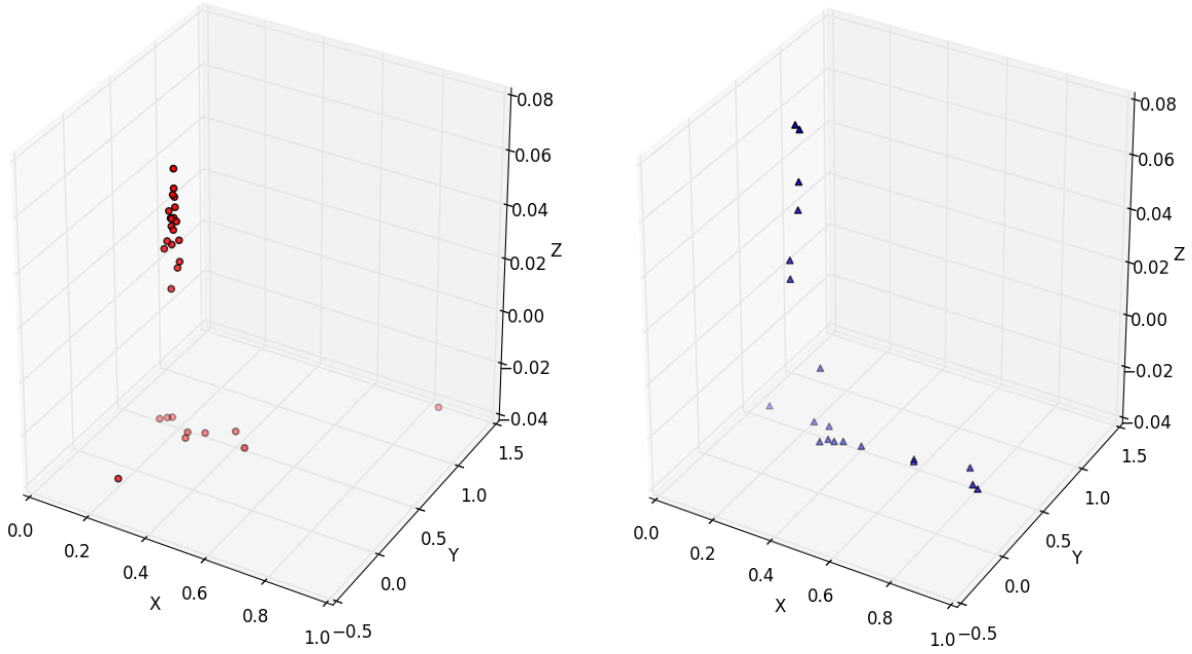
Manipulator robot parameters

(Parameters determined at the start of a particle trace)
Model:
link density: $\{6\times$arm, $1\times$hand, $4\times$fingers$\}$
link length: $\{6\times$arm, $1\times$hand, $4\times$fingers$\}$
link radius: $\{6\times$arm, $1\times$hand, $4\times$fingers$\}$
joint axis (conical error): $11\times$revolute joints
Environment:
contact friction, contact restitution, contact model
Initial state:
$q_1 \cdots q_{11}, \dot{q}_1 \cdots \dot{q}_{11}$
Other:
control lag

(Parameters determined during particle trace execution)
Encoder noise:
$q_1 \cdots q_{11}, \dot{q}_1 \cdots \dot{q}_{11}, \ddot{q}_1 \cdots \ddot{q}_{11}$
Sensed actuator torque noise: $u_1 \cdots u_{11}$
Other: control lag jitter

Figure 10: ***This box describes relevant scenario information to execute the particle traces approach for an 11 DOF simulated manipulator with a 4 DOF gripper performing a picking task with direct and indirect operational space gripper trajectory plans.***

Results Examples of expected and unexpected behaviors using each trajectory are depicted in Figures 8 and 9, respectively; these provide a visual representation of a particle trace for a manipulator within this framework.



(a) ***Policy A: More Robust*** **(b)** ***Policy B: Brittle***

Figure 11: ***Final position of the ball after the pick behavior following Path A or B. All units are in meters. The z-axis is vertical (i.e., positive values of z correspond to up with respect to gravity).***

Points along the bottom of the plot in Figure 11 (on the $z = -0.04$m plane) correspond the ball resting on the ground; these states fail the task objective. All points not resting on the plane correspond to the ball being held by the gripper and satisfy the task objective. Kinematic uncertainty of the manipulator model results in a large variation in the position of the grasped ball after the task is completed successfully; this results in different gripper positions for the same desired final joint configuration (i.e., the final ball positions—held by the robot's gripper—for successful traces are not all overlapping). Figure 12 shows an image of all randomly perturbed manipulator robots (one per particle) at the same joint configuration. Kinematic differences between perturbed robot models resulted in disparate gripper positions for one set of joint angles.

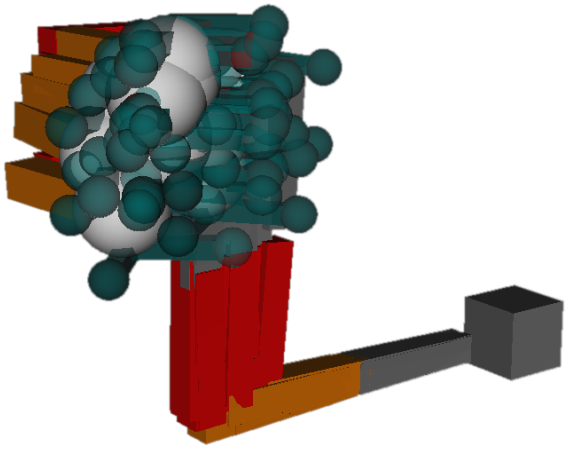


Figure 12: ***All randomly sampled particles of the manipulator robot with the same configuration space state.***

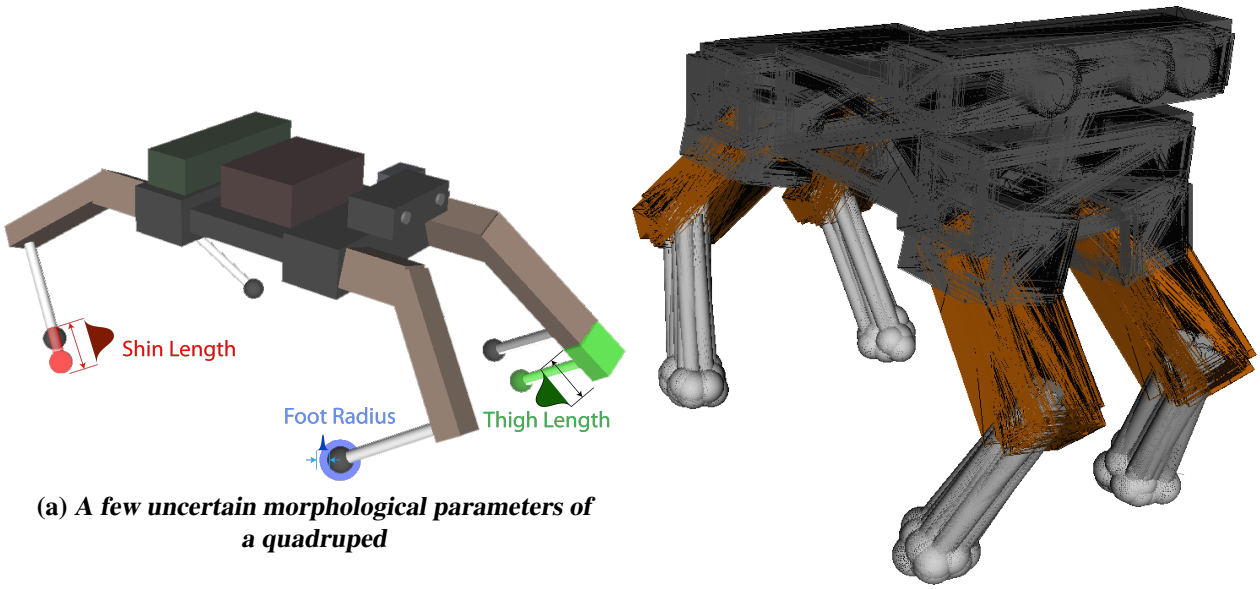
Reaching Policy A resulted in an 88% success rate while Policy B successfully completed the picking task 63% of the time (a 40% performance differential). A finger inadvertently tapping the sphere and causing it to roll out of reach was a typical cause of failure for Policy B. Figure 11 shows the final position of the ball in each of the particle traces. This result is consistent with the premise behind the work in Rombokas et al. (2012) (i.e., the approach to a grasp has a strong determination of grasp success *in situ*); the particle traces algorithm discovered that the direct approach control policy (Policy A) was more robust to model and state uncertainty.

Discussion of Results The results of this section reflect the expectation that small "errors" (i.e., state estimates, control, modeling parameters, etc.) can have large effects on manipulation. The relative particle trace success rates between the two reaching policies reveals that the superficially successful policies differed in their robustness; one control policy (Policy A) exhibited significantly more particle traces that satisfied the task objective, indicating a more robust policy. Simulation results comparing the control policies for this task indicate that Policy A might be a preferable control policy for *in situ* execution; it exhibited the intended behavior in a greater number of traces, indicating that it might be a more robust control policy. The success rate of a policy indicates the robustness of the policy when executed *in situ* (or, at least, the relative robustness to other policies), but testing that hypothesis was not the focus of this experiment. A validation experiment was performed for the particle traces approach *in situ* on a different scenario (§4.4).

4.3.2 Quadruped: detecting grazing bifurcation Detecting and avoiding catastrophic failure cases might not be enough to predict successful behavior; indeterminate behavior must be avoided as well. The quadruped experiment will show that bifurcating events that do not necessarily translate to failure yet might still result in unpredictable behavior *in situ* can be detected by observing the telemetry of particle traces. Throughout all particle traces of this curb traversal experiment, the robot remained upright and stable but still exhibited undesirable divergent behavior.

A twelve-jointed, floating-base quadrupedal robot was simulated from an initial position next to a curb obstacle. The quadruped was directed to step over the curb. Differing step heights, but otherwise identical locomotion control policies were used to control the quadruped. Figure 16 shows a time-lapse depiction of the diverging behaviors that the robot might exhibit. The robot collides with the curb if the foot is not lifted high enough to clear the obstacle.

Grazing bifurcation might occur if the step height is approximately equal to the curb height: small changes in initial conditions, modeling parameters, or sensing (of, e.g., curb geometry) would determine whether or not the robot would strike the obstacle *in situ*. Four trials were run, each of which used one of four preset gait control policies that attempts one, two, three, and four centimeter step heights. The curb height was fixed at three centimeters.



(a) ***A few uncertain morphological parameters of a quadruped***

(b) ***Sixty-four overlaid quadruped particles with randomly sampled morphological parameters***

Figure 13: ***A depiction of the probabilistic geometric parameters of a legged robot: shin length, thigh length, and foot radius.***

Control policy: gait parameters Planning and control for the simulated robot was generated by the PACER planning and control software (see Chapter 9). Gait parameters are defined in Figure 1.

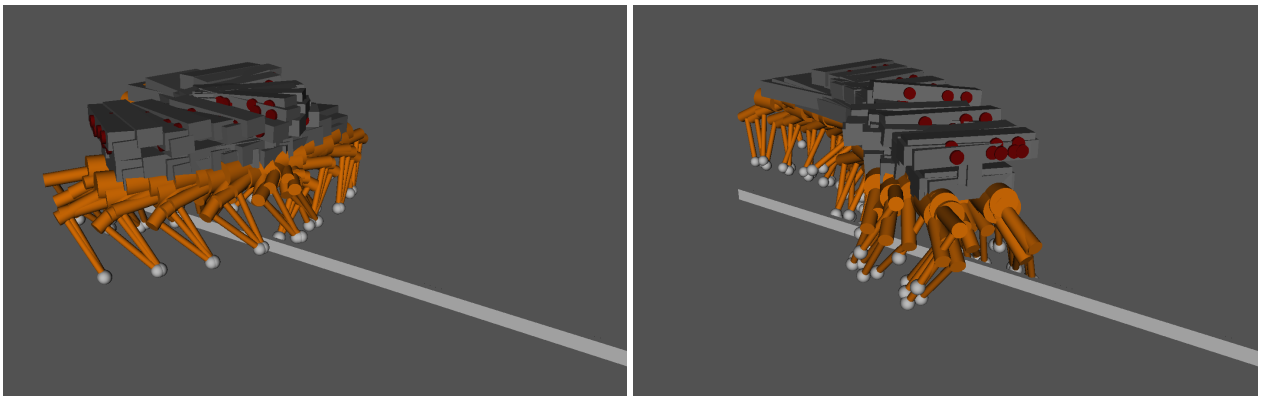
Quadrupedal Gait Parameters

Parameter (unit)	Value
forward velocity (cm/s)	20
yaw velocity (rad/s)	-0.5
base linear offset (cm)	$\{0, 0, 0.9 \times$ min-leg-length$\}$ (see Table 6)
base orientation offset (rad)	$\{0, 0, 0\}$
step height (cm)	$h_s \in \{1, 2, 3, 4\}$ (see Figure 15)
stance length (% base length)	120
stance width (% base width)	200
gait duration (sec)	0.3
liftoff timing (% gait duration)	$\{25, 75, 0, 50\}$
duty factor (% gait duration)	$\{75, 75, 75, 75\}$

Table 1: *Gait parameters for the walking task performed by the simulated quadruped when attempting to step over a curb. See Chapter 9 for a description of these parameters.*

Task Objective & Requirements The ability of the quadruped to traverse an obstacle is assessed using an open loop gait control policy. The goal for this scenario was for the robot to walk over the

obstacle. This goal is identified with a task objective that all four feet of the robot are on the far side of the obstacle after the full 1.5 second policy duration ($t^{\max} = 1.5$). A few safety requirements and early warnings of objective failure were added to the list of task requirements as well: for safety, a requirement failure was triggered if the virtual robot self-collided or if the robot torso came into contact with anything (indicating either self collision or that it somehow collapsed without falling over); task requirements that the robot never strikes the side of the curb while walking and that the robot remain upright throughout the test were added for saving computation with early task failure indication. A foot striking the side of the curb would indicate that a control policy did not lift the foot high enough to step over the curb. See Figure 15 for a list of task objectives and requirements for this scenario.



(a) ***High step: the quadruped successfully crosses the curb***

(b) ***Low step: the quadruped stubs its toe on the curb***

Figure 14: ***A time-lapse of the virtual* LINKS *robot walking over or running into a curb obstacle with high (4cm) and low (1cm) step heights. The robot that is not able to walk across the curb fails at completing the task objective.***

Particle Parameters The virtual quadruped robot has a floating base and twelve revolute joints, three per limb. 175 parameters determine geometric, kinematic and dynamic properties of the robot and its environment; these parameters are listed in Figure 15.

Quadruped walking over a curb

Control Policy: *gait control policy*

Gait with step heights (cm): $h_s \in \{1, 2, 3, 4\}$

Task Description

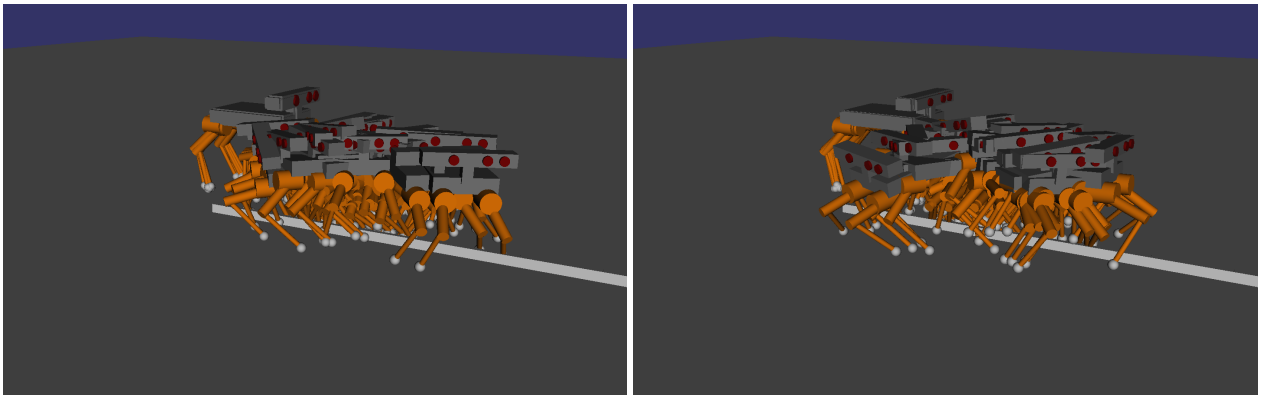
Objective:
Robot is upright at time $t^{\max}$
All feet on far side of curb at time $t^{\max}$
Requirements:
Robot remains upright
Foot does not strike curb
Body does not contact environment
Non-adjacent robot links do not contact one-another

Quadruped robot parameters

(Parameters determined at the start of a particle trace)
Model:
link density: {1×base, 4×hip, 4×thigh, 4×shin, 4×foot}
link length: {4×hip, 4×thigh, 4×shin}
link radius: {4×hip, 4×thigh, 4×shin, 4×foot}
joint axis (conical error): 12×revolute joints
Environment:
contact friction, contact restitution, contact model
Initial state:
$x, y, z, \psi, \phi, \theta, q_1 \cdots q_{12}, \dot{x}, \dot{y}, \dot{z}, \dot{\psi}, \dot{\phi}, \dot{\theta}, \dot{q}_1 \cdots \dot{q}_{12}$
Other:
control lag

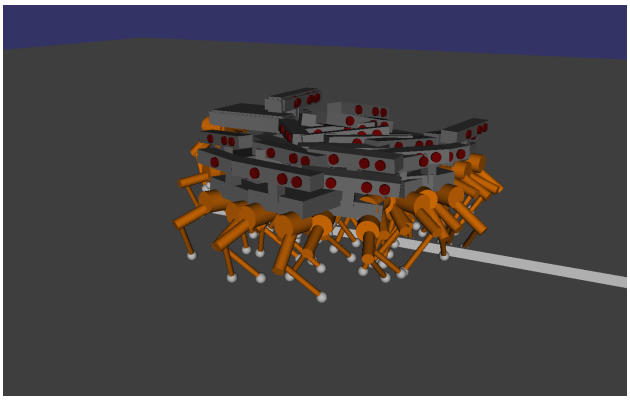
(Parameters determined during particle trace execution)
Encoder noise: $q_1 \cdots q_{12}, \dot{q}_1 \cdots \dot{q}_{12}, \ddot{q}_1 \cdots \ddot{q}_{12}$
Force sensor noise: $u_1 \cdots u_{12}$
IMU noise: $\ddot{x}, \ddot{y}, \ddot{z}, \ddot{\psi}, \ddot{\phi}, \ddot{\theta}$
GPS noise: x, y
Magnetometer noise: ψ, ϕ, θ
State Estimation noise: $z, \dot{x}, \dot{y}, \dot{z}, \dot{\psi}, \dot{\phi}, \dot{\theta}$
Sensed actuator torque noise: $u_{\text{des},1} \cdots u_{\text{des},12}$
Other: control lag jitter

Figure 15: ***This box describes relevant scenario information to execute the particle traces approach for the simulated quadruped performing the walking over a curb task with variable step height.***

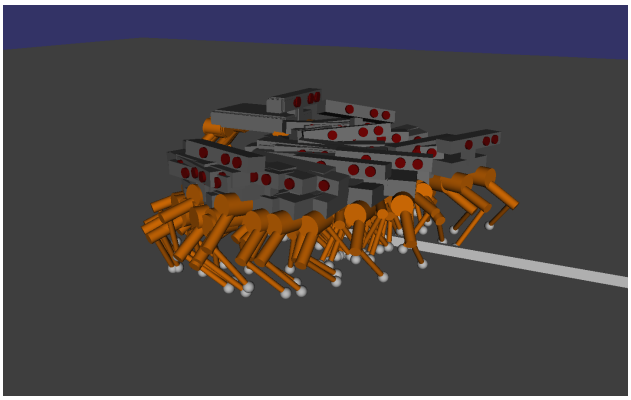


(a) ***Step height: 1 cm***

(b) ***Step height: 2 cm***



(c) ***Step height: 3 cm***



(d) ***Step height: 4 cm***

Figure 16: ***A time-lapse of the virtual*** **LINKS** ***robot walking over or running into a curb obstacle given various step heights. Each particle trace is rendered along one-second intervals. Higher step heights result in the robot standing fully on the near side of the curb; short step heights fail to cross the curb and exhibit final robot configurations with one or more legs on the far side of the curb.***

Results Traces following the three cm step-height control policy fall into two distinct classifications of behavior, corresponding to (1) passing the curb obstacle and (2) running up against the curb obstacle and not crossing; both behaviors pictured in Figure 14 are present in Figure 16c. The presence of these two groups of final states exhibited when following the 3cm control policy indicates that the particle parameters determine whether or not the robot is able to traverse the obstacle. The sampling strategy uncovered the grazing bifurcation that led to these divergent behaviors as the variance of the green paths in Figure 17 indicates. It was concluded from this information that the three cm step-height control policy is brittle. It would be expected that the robot's *in situ* performance would be difficult to predict under this policy because the robot's vir-

tual behavior is sensitive to modeling and estimation uncertainty. In contrast, the four cm step height policy allowed the quadruped to step over the curb for almost all traces and the one and two cm step height policy caused the robot to collide with the curb in all traces. It is reasonable to expect that the real robot would behave predictably (usually succeeding or failing at the task) using these policies and would behave successfully *in situ* with a sufficiently high step height. It is possible that a specialized algorithm could be devised to calculate an ideal step height, possibly even for terrain composed of varied irregular obstacles. However, the particle traces approach is not specialized; Section 4.3.1 showed how this same algorithm and scoring technique was used without modification to assess the robustness of a control policy for a manipulator. The particle traces approach offers the use of computational resources and simulation as a general solution to plan selection in complex robot scenarios.

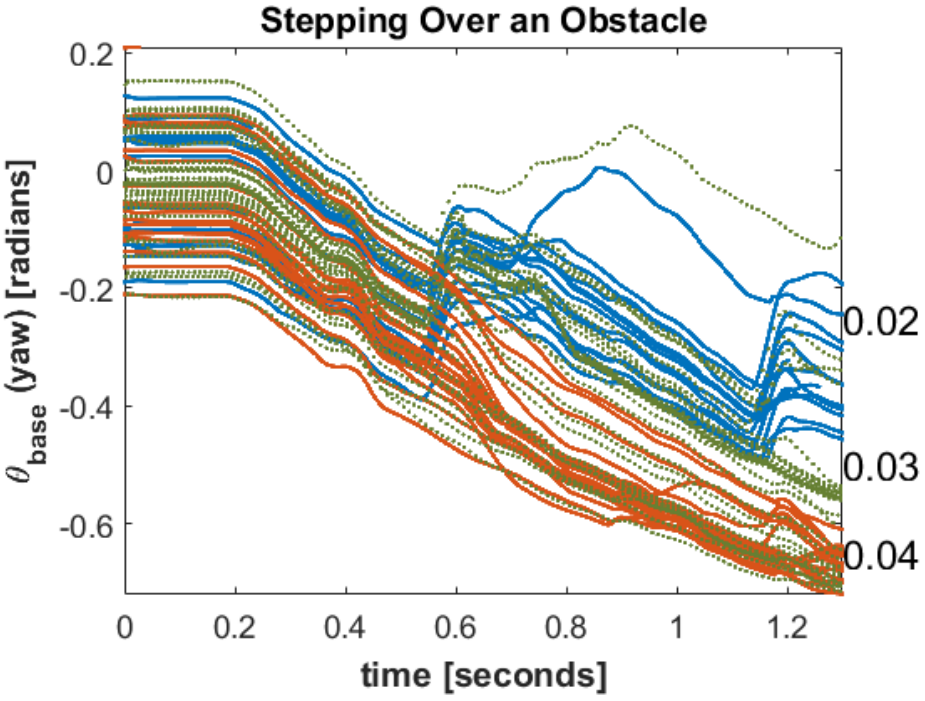


Figure 17: ***Virtual quadrupedal robot base yaw when turning into a 3 cm tall curb obstacle. Each line represents a particle, and each color represents a policy. Red particle traces followed a 4 cm step height policy (marked as 0.04 m on plot) step over the curb and continue to turn. Blue particle traces followed a 2 cm step height policy (marked as 0.02 m on plot) strike the curb and are prevented from turning. Green, dotted particles traces followed a 3 cm step height policy (marked as 0.03 m on plot), where step height matches the curb height (3 cm), exhibit divergent non-failing behavior by only occasionally striking the obstacle.***

Discussion of Results Simulating many particle traces can generate huge amounts of telemetry data. One way to manage the data output from the particle traces approach is to mine information about the robot's telemetry efficiently. All of the particle traces from this curb traversal experiment can be classified into successful and failing groups by observing a short sequence of events while the robots are starting to step over the obstacle: the contact sequence { RH–ground, LH–ground, RF–ground, LF–ground } indicates that the robot stepped over the obstacle successfully, and the contact sequence { RH–ground, LH–ground, RF/obstacle, RF–ground, LF–ground } indicates that the robot did not step high enough and struck the curb. This result indicates that a witness function tracking the robot's distance from striking the curb might be helpful as an indicator of robustness rather than relying on observing the final states of the robot. However, monitoring such witness functions presents a more computationally intensive task (see Chapter 5 for a deeper look into using witness functions as an indicator of robustness).

***4.3.3 Quadruped: Verifying the stability of gait transition timing* online** Chapters 3 and 4 describe, implement and test a statistically based predictive framework for underconstrained, limbed robots. Although the implemented approaches demonstrate some success *in situ* (§4.4), they are predicated on the prior knowledge of the robot's environment and some measure of that system's uncertainty. If virtual falsification is used *online* then details about the robot's environment and the arbitrary task it must perform must be known ahead of time—a strong assumption or new information gained from the robot's sensors must be incorporated into the particle trace computation in real-time. Model predictive control (as opposed to optimal control) avoids pre-computation, and can use the current model, initial state, and environment. This section presents a framework for the parallelized Monte Carlo method toward online model predictive control.

Task For this virtual trial, LINKS walks forward while following a sinusoidal path between waypoints (drawn in Figure 19a). During the second gait cycle, at the *gait transition point* the robot's controller instantly switches from the *starting gait* parametrization to the *new gait*. The robot then continues to locomote with the *new gait* parameterization for two seconds or until failure. Discovering a stable transition between two arbitrary gait policies is an open problem in robotics and is

the target of ongoing study (Gehring et al., 2013). The timing of the *gait transition point* differed between control policies. Parameters for the starting and new gaits are defined in Figure 1.

Starting Gait Parameters: Walk

Parameter (unit)	Value
max forward velocity (cm/s)	10
max strafe velocity (cm/s)	5
max rotational velocity (rad/s)	0.5
base linear offset (cm)	$\{0, 0, 0.75 \times$ min-leg-length$\}$ (see Table 6)
base orientation offset (rad)	$\{0, 0, 0\}$
step height (cm)	3
stance length (% base length)	120
stance width (% base width)	120
gait duration (sec)	0.3
liftoff timing (% gait duration)	$\{25, 75, 0, 50\}$
duty factor (% gait duration)	$\{75, 75, 75, 75\}$

New Gait Parameters: Trot

Parameter (unit)	Value
max forward velocity (cm/s)	50
max strafe velocity (cm/s)	10
max rotational velocity (rad/s)	1
base linear offset (cm)	$\{0, 0, 0.75 \times$ min-leg-length$\}$ (see Table 6)
base orientation offset (rad)	$\{0, 0, 0\}$
step height (cm)	3
stance length (% base length)	120
stance width (% base width)	120
gait duration (sec)	0.3
liftoff timing (% gait duration)	$\{25, 75, 75, 25\}$
duty factor (% gait duration)	$\{60, 60, 60, 60\}$

Table 2: *Gait parameters for the gait-switching task performed by the simulated quadruped.*

The virtual quadruped robot is the same as the previous curb traversal and validation scenarios; particle trace parameters specific to this scenario are listed in Figure 15.

Quadruped gait-transition timing

Control Policy: *gait control policy*

Transition from *starting gait* parametrization to *new gait* at *gait transition point* (% gait duration) *gait transition point* $\in \{0, 25, 50, 75\}$

Task Description

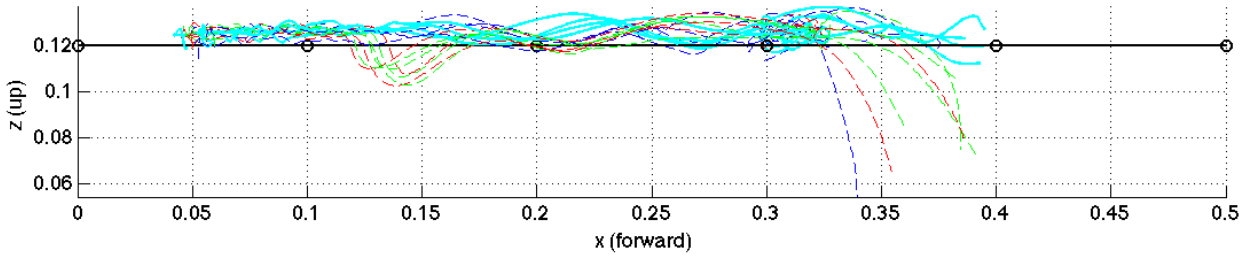
Requirements:
Completes two seconds of the *new gait* after the gait transition, without falling over.
Pitch limit (stability)
Roll limit (stability)

Quadruped robot parameters

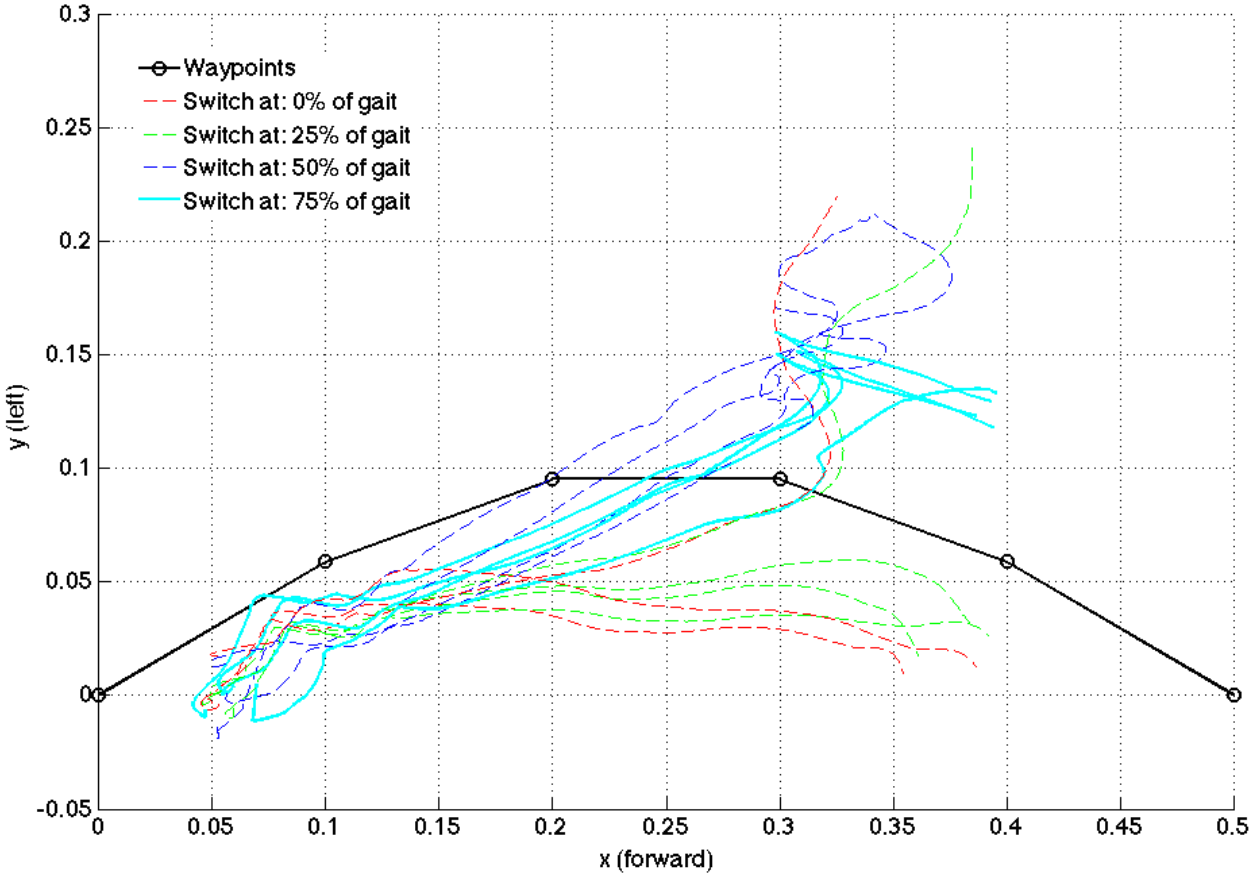
same as in Figure 15

Figure 18: ***This box describes relevant scenario information to execute the particle traces approach for simulated quadruped performing the quadrupedal gait-transition-timing task.***

Results Results for the gait switch procedure using the online particle traces approach are plotted in Figures 19 and 20.



(a) ***Side View***



(b) ***Top View***

Figure 19: ***Top and side views of the four particle traces for each candidate control policy. Policies that fail are drawn with dotted lines; successes are drawn with bold lines. The waypoints marking the robot's path are marked as black circles. The robot moves in the*** $+x$ ***direction.***

Using only four particle traces (coinciding with the four cores of the testing machine) a few important aspects of the tested gait transition timings were observed: (1) a transition time at 50% of the way through the starting gait led to a well clustered output of final states, but no traces satisfied the task objective. (2) transition times at 0% and 25% each exhibited one divergent trace

falling to the left of the desired path, and the rest fell to the right; none of the traces exhibited the desired behavior after the gait transition was activated. (3) a transition at 75% of the way through the starting gait exhibited successful behavior for all traces. The output from the particle traces approach indicated that the policy with the 75% gait transition point would be most reliable *in situ* due to the low variance of the final states of the traces and a high rate of success.

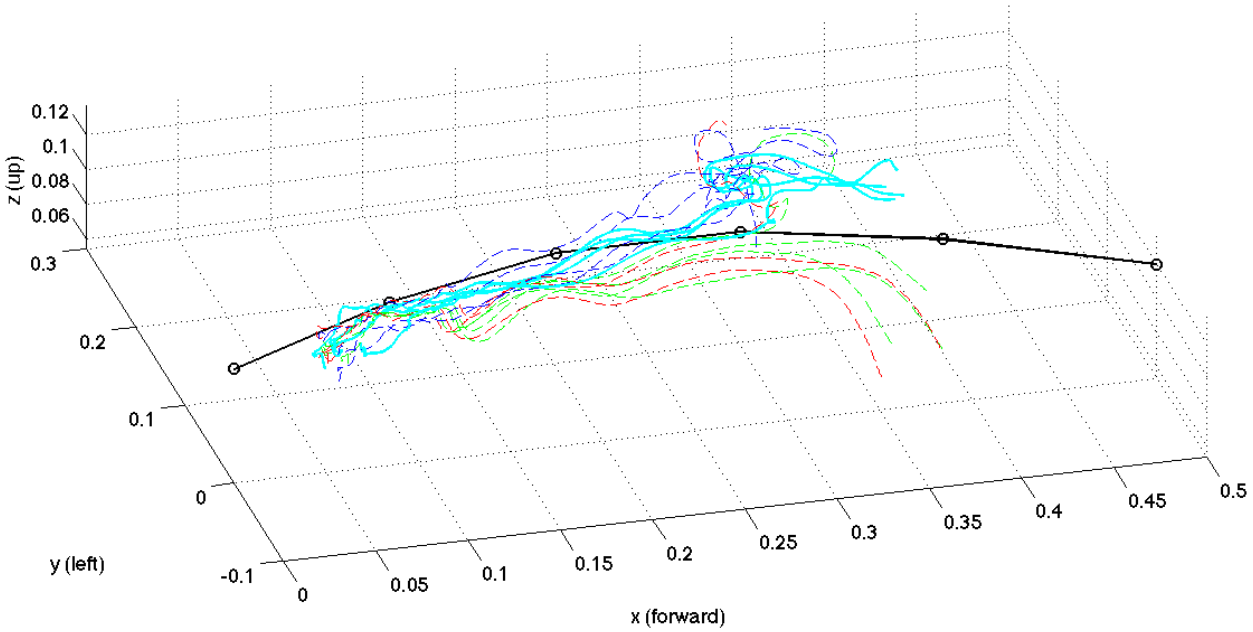


Figure 20: ***An isometric view of the four particle traces for each candidate control policy. Policies that fail are drawn with dotted lines; successes are drawn with bold lines. The waypoints marking the robot's path are marked as black circles. The robot is moving in the*** $+x$ ***direction.***

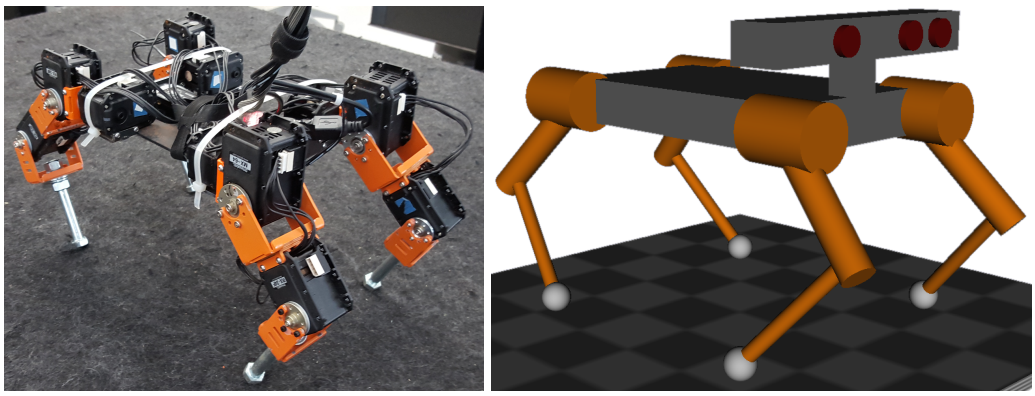
The preliminary results presented in this section demonstrate that just a few particle traces can detect disparate behaviors for a single control policy and even select a more robust choice among a group of candidate policies. The following section describes and illustrates an implementation of the particle traces approach for such *online* model predictive control.

4.4 Validation of particle traces approach *in situ*

This section describes validating the particle traces approach on a quadrupedal robot *in situ*. The results from the experiment in this section demonstrate that the particle traces approach can be used to detect whether a locomotion control policy might cause a robot to fall; divergent behavior

exhibited by the robot controlled when following a certain gait policy can be detected in simulation using virtual falsification with even a small number of sampled particles.

4.4.1 Robot I focus on the correlation between simulated and *in situ* behavior for a scenario that *should* be modelable using fast simulation tools. This issue is important because one can only expect grazing bifurcations located within simulation to be informative if there is some correlation between simulated and *in situ* behavior. The robot used in physical trials, LINKS, is an 18 degree-of-freedom (12 actuated) quadruped robot constructed from Dynamixel actuators and steel links (see Figure 53a). Base orientation was recorded by an IMU that produces samples at 100 Hz. Modeling inaccuracies and measurement errors on even such a small robot are legion and would include the rigid body assumption, gear backlash, communication delay, IMU sensing delay, the rigid contact assumption, and back EMF.



(a) ***The* LINKS *robot* in situ**

(b) ***A physical model of* LINKS *in the* MOBY *simulator***

Figure 21: ***The* LINKS *robot in position to begin a walking experiment.***

Dynamics model The virtual quadrupedal robot was modeled using a box for the base link inertia and geometry, cylinders for the limb link inertias and geometries, and spheres for foot inertias and geometries. Modeling parameters for the virtual quadruped were set from measurements collected from the LINKS robot (Figure 21a). There are no compliant elements in the structure of LINKS (unless one counts the transmission), which should allow it to be well modeled as a multi-rigid body with high accuracy.

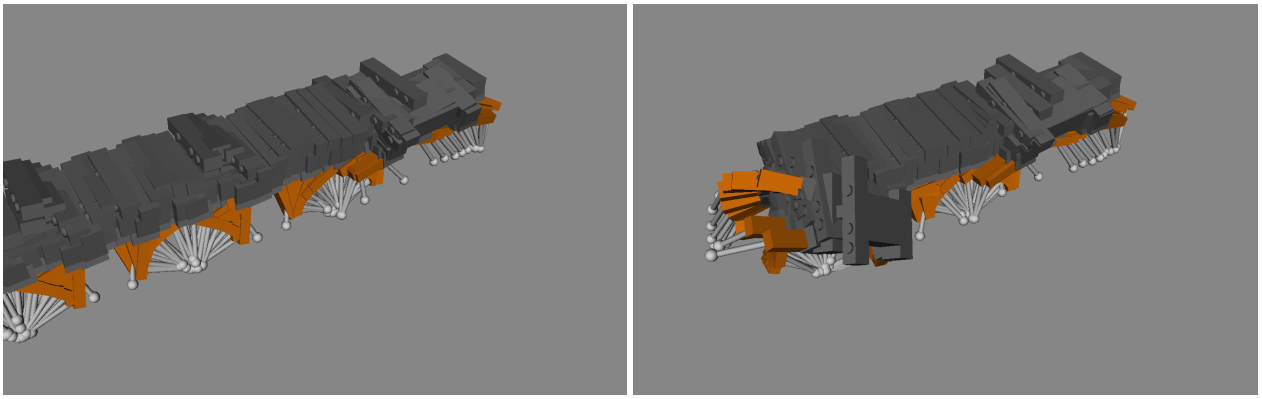
*4.4.2 **Control policy: gait parameters*** A single gait parameter was adjusted (gait period duration) how it affected the behavior of the robot was observed over the course of the experiment. Gait period duration was adjusted from an empirically observed open-loop stable gait (0.6 seconds per gait cycle), upward to a value where definite failure had previously been observed (1.5 seconds per gait cycle). Each particle trace was executed over 20s of virtual time or until a fall, and LINKS was permitted to walk for 20s of wall time. Twenty seconds was chosen as a sufficiently long test duration to excite any destabilizing events; too little time would not stress the control policy enough and too much would be both unnecessary and time consuming from a computation perspective. The gait parameters for this task are defined in Figure 3.

Quadrupedal Gait Parameters

Parameter (unit)	Value
forward velocity (cm/s)	20
yaw velocity (rad/s)	0
base linear offset (cm)	$\{0, 0, 0.9 \times$ min-leg-length$\}$ (see Table 6)
base orientation offset (rad)	$\{0, 0, 0\}$
step height (cm)	3
stance length (% base length)	120
stance width (% base width)	200
gait duration (sec)	$d_g \in [0.6, 1.5]$ (see Figure 23)
liftoff timing (% gait duration)	$\{25, 75, 0, 50\}$
duty factor (% gait duration)	$\{75, 75, 75, 75\}$

Table 3: *Gait parameters for the walking task performed by the simulated quadruped. See Chapter 9 for a description of these parameters.*

Task Objective & Requirements This example was meant primarily to assess the stability of an open loop locomotion control policy: keeping the robot oriented in a specific yaw direction was not an objective of this task. Figure 22 demonstrates what a failure (i.e., destabilization, then a fall) might look like for this task. Task requirements are identical to the previous curb traversal example. See Figure 23 for a list of task objectives and requirements for this scenario. The orientation of the robot base was recorded to detect failure during *in situ* trials. A configuration was labeled a fall if the "up" axis in the robot's local frame exceeded $\frac{\pi}{2}$ radians displacement from "up" in the global frame.



(a) ***Quadruped successfully walking for several steps***

(b) ***Quadruped destabilizing during its third gait cycle***

Figure 22: ***A time-lapse of the virtual* LINKS *robot walking in a straight line with a single gait duration (1.1s). Some traces fail due to modeling uncertainty.***

Particle Parameters The virtual quadruped robot is the same as the previous curb traversal scenario; particle parameters specific to this scenario are listed in Figure 23.

Quadruped walking on a half-space

Control Policy: *gait control policy*

Gaits with *gait cycle duration* (seconds): $d_g \in [0.6, 1.5]$

Task Description

Objective:
Robot is upright at time $t^{\max}$
Requirements:
same as in Figure 15

Quadruped robot parameters

same as in Figure 15

Figure 23: ***This box describes relevant scenario information to execute the particle traces approach for the simulated quadruped and the* LINKS *robot* in situ *performing the walking on a plane task with variable gait cycle duration.***

4.4.3 Results A time-lapse depiction of each control policy in this scenario is presented in Figure 27. *The results from this experiment indicate that when all particle traces in a trial correspond*

to walking without a fall, LINKS *does not fall* in situ. Gait period durations between 0.6 and 1.0 exhibit robustness to modeling infidelities, variability in initial conditions, and errors in state estimates. When some particle traces exhibited falling behavior for a control policy, the LINKS robot walked several steps *in situ* before falling. When all particle traces exhibited falling the LINKS robot fell on its first step *in situ*; Figure 24 shows this correlation between the duration walked before a fall *in situ* and the average duration before a fall for all particle traces with respect to the gait period duration of the locomotion control policy.

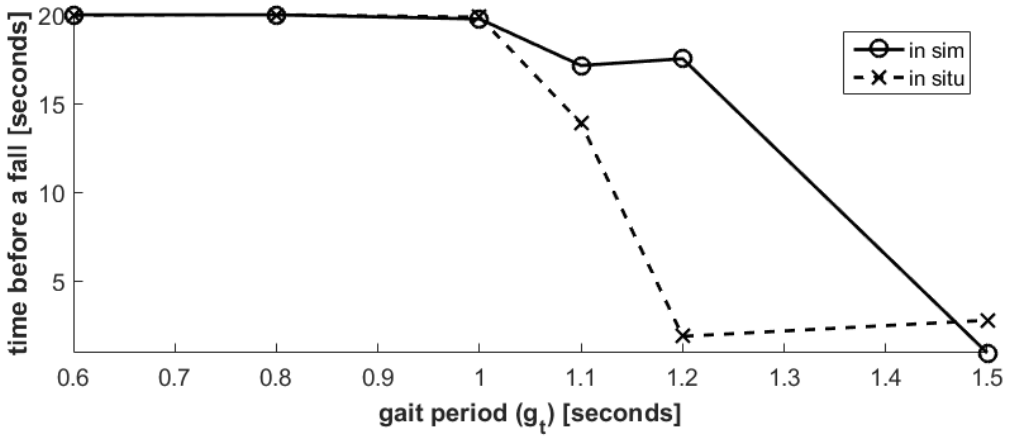
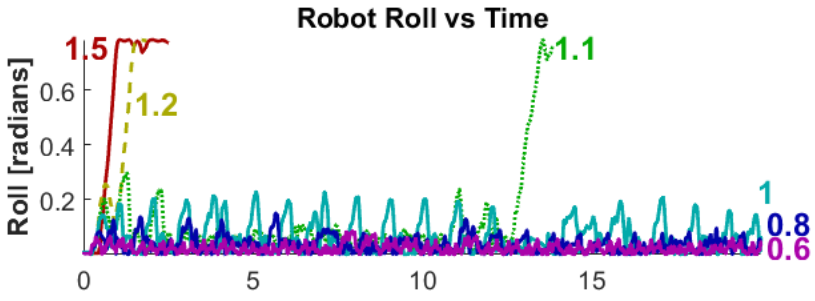
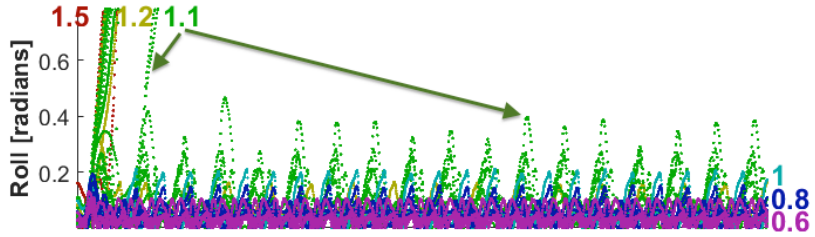


Figure 24: ***Duration of time until a fall of the locomoting robot plotted with respect to the gait period duration parameter.***
(× mark, dotted line) Duration of wall time until a fall of the* LINKS *robot.
(○ mark, solid line) Average duration of virtual time until a fall for all particle traces.

Figure 24 plots the roll orientation data of the physical and virtual robots using the several different locomotion control policies. There is a qualitative agreement between these plots: the magnitude of variation in roll grows as the gait period duration lengthens. Both simulated and *in situ* robots fell at a 1.1 second gait period duration or longer. The particle traces approach was able to predict that the control policy was brittle using only information from simulation; observations from situated testing support the predictions made by the particle traces algorithm.



(a) in situ



(b) in sim

Figure 25: ***Roll orientation data for the walking quadruped robot. Each line is labeled with its corresponding value of the gait period duration for each policy. The dotted line for the simulation data plots four overlaid sets of data for each control policy.***

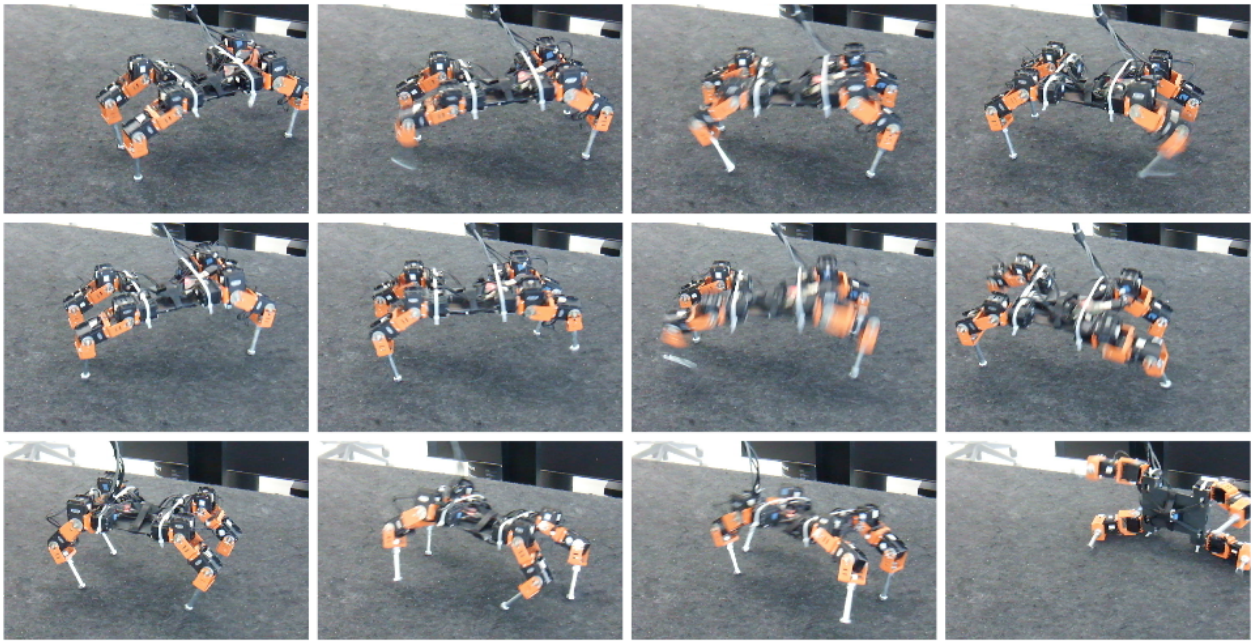
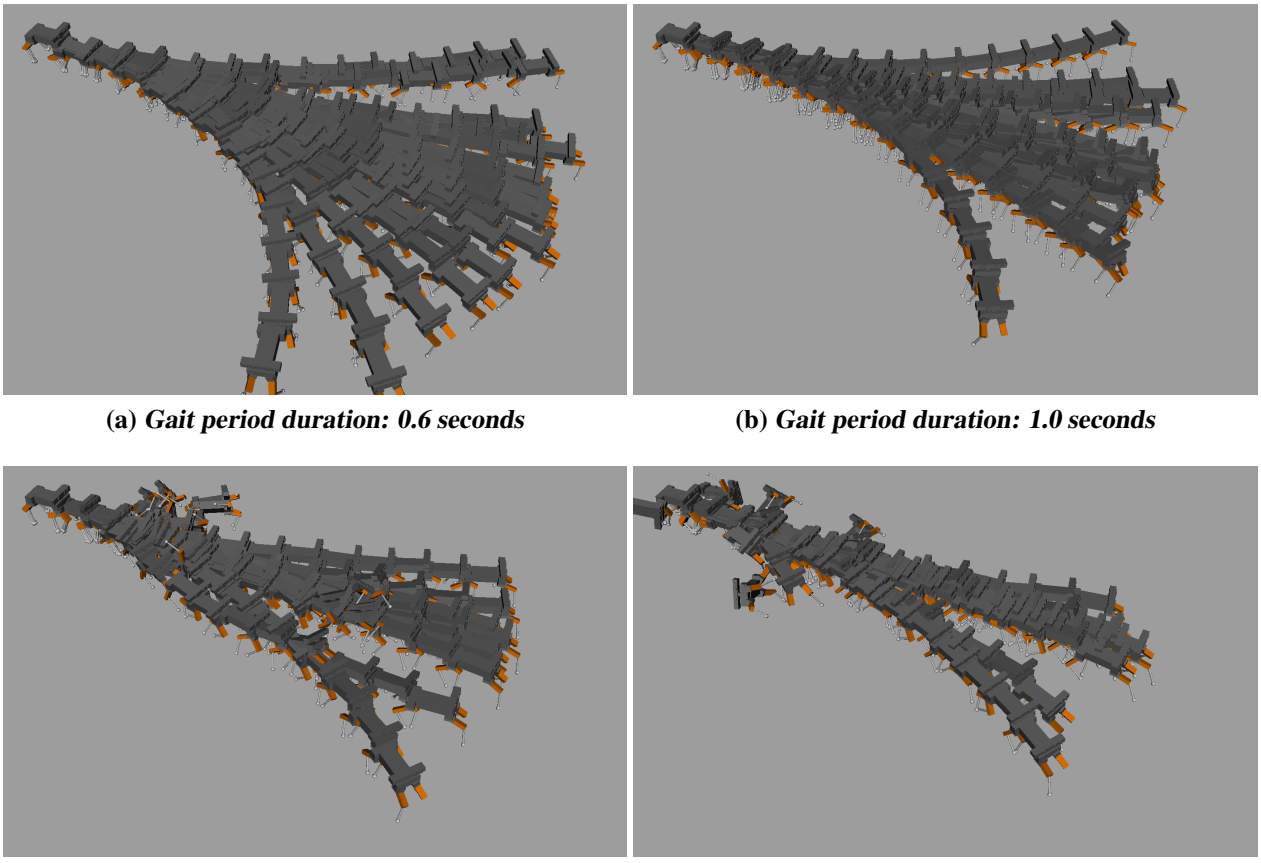


Figure 26: ***A two second time-lapse of*** **LINKS** ***walking with a gait period duration of: 0.6 seconds (Top); 1.0 seconds (Middle); 1.5 seconds (Bottom). The robot became progressively less stable as the gait period duration increased.***

Bifurcation versus scattering The magnitude of the effects of "scatterers" and "bifurcators" (Section 4.3.2) on the robotic system appeared to vary as a function of the "gait period duration". Comparing the behavior of the control policy with a short gait period duration (0.6s) to a moderate gait period duration (1.0s), the robot with the faster gait period took a greater number of steps—and experienced more impacts—over the same duration of time; the shorter gait period resulted in a larger scattering effect over the same duration of locomotion, as evinced by grater dispersion in the region of final positions. Once the gait period duration was increased beyond 1.0s, some traces exhibited divergent behavior to a fallen-over configuration. Increases in duration beyond 1.0s correlated with a stronger bifurcation effect and a greater number of particle traces that corresponded to undesirable behavior. A continuation of the scattering effect was observed in longer duration gaits as well. Both effects were observed between the four panes of Figure 27: (1) Figure 27a exhibits a large scattering effect; (2) Figure 27b exhibits a moderate scattering effect; (3) Figure 27c exhibits a lower scattering effect, but some traces exhibit divergent behavior to an absorbing state (i.e., falling-over), resulting in failure to produce the desired behavior; (4) Figure 27d exhibits the lowest scattering effect, but also the highest rate of failure.



(a) *Gait period duration: 0.6 seconds*

(b) *Gait period duration: 1.0 seconds*

(c) *Gait period duration: 1.1 seconds*

(d) *Gait period duration: 1.5 seconds*

Figure 27: ***A twenty second time-lapse of*** **LINKS** ***walking with different gait period durations. Each particle trace is rendered along one-second intervals.***

4.5 Conclusion

The experiments in this chapter have shown that it is possible to locate seemingly hard to detect grazing bifurcations (i.e., ones that lie in small volumes in state space) for high dimensional systems with relatively few particle traces. These results emphasize the need for simulated testing involving multiple samples and perturbed robot models; without the information provided by multiple samples it would be difficult to make predictions of brittle robot behavior. The 1.1s gait period duration control policy in Section 4.4 led a robot to fall half-way through a test *in situ*; that control policy only led to particle traces that either exhibited a fail immediately or not at all. Due to the brittleness of the control policy a single virtual test might have exhibited either of these behaviors. The consensus of multiple particle traces was needed to determine the trustworthiness of simulated

results and the predictability of a robot's behavior, before moving to physical experiments. The results in this chapter indicate that such brittle policies can be identified quickly (i.e., with a small number of samples).

This chapter presented experiments that demonstrate the adaptability of and then validated the particle traces approach from Chapter 3. The next chapter, Chapter 5 presents an algorithmic approach toward mitigating the divergent behaviors exhibited by the robots in this chapter.

5 Computer-aided robot improvement

Design of legged, humanoid, and other dynamically moving limbed robots is particularly challenging in comparison to many engineering disciplines because robot designs have yet to converge. In automotive engineering, for example, the basic design principles of a car are presupposed: four wheels, suspension, powerplant, transmission, disc brakes, etc., and automotive technology is able to progress by making small modifications in this design space. Designing limbed robots, on the other hand, requires simultaneously optimizing hardware and software. These factors are interdependent; for example, reducing contact compliance through increased mechanical stiffness requires a high bandwidth control system that can limit interaction forces.

The work in this chapter was initiated by the hypothesis that for limbed robots, as with automobiles, peak dynamic performance tends to occur as the hardware components and control strategy push against limits. Auto racing pushes tires to their traction limits, tunes engines and transmissions to optimize the powerband, and tweaks suspensions to maximize the tire contact patch. In terms of "controllers", drivers practice race tracks repeatedly to tune their models of tracks and cars toward minimizing lap times.

Observations made while working to test this hypothesis indicated that optimizing robot designs often pushes the robot into an infeasible parameter space. To make a legged robot run, for example, acceleration commands usually push against torque limits, velocity commands push against maximum actuator speeds, strides tend to push against joint limits (toward giving a limb time to accelerate to maximum speed), and foot trajectories come close to scraping the ground. Traces exhibiting behaviors that fail to satisfy a task objective or fail a task requirement in Chapter 4 can be attributed to the robot encountering one of these limits, leading to an unexpected and possibly non-recoverable perturbation to the state of the robotic system or its environment (discussed in Section 3.6.1). Correspondingly, this work investigates a method to adjust a robot's physical design by altering its "parameter space" away from values that cause the robot to encounter these limits. The model updating approach presented in this chapter is analogous to projecting an "infea-

sible point" in the robot control policy and design parameter space (that nevertheless corresponds to a good objective function value) to a nearby feasible point, a common strategy for constrained optimization.

Section 5.1 describes the constituent parts of the virtual model modification approach presented in this chapter. Section 5.1.1 describes how task performance and divergent behavior relate to a robot encountering a physical limitation of its hardware. Section 5.1.2 then details a robot's proximity to violating one of these physical limitations can be monitored by implementing a *witness function*. Section 5.2 then describes the implementing of an algorithm applying the proposed model modification approach; it explains how the algorithm detects stability-related divergent behaviors (§5.2.2) and how a robot's morphological parameters relate to its task performance (§5.2.1).

5.1 Approach

Model predictive control and numerical optimal control techniques in robotics have received considerable attention because of their great promise: specify an objective function, constraints, and models and—assuming the computational intractability issues can be resolved—perfect controllers result. That possibility is much more palatable than the status quo, as those experienced with writing software to control robots in unstructured, uncertain environments can attest. That attention supposes that objective functions for robots can be readily devised. This chapter considers both a slightly different problem (assume that the hardware is modifiable, i.e., that the design is fluid) and a different solution: use computer aided design to adjust parameters of a robotic system (hardware design in this chapter, but hardware and software parameters generally) in order to respect constraints affecting the performance of the robot.

The work in this section builds upon research in evolutionary robotics, physical simulation, rapid prototyping, and operational space locomotion. This section builds upon this work by providing an interactive robot design suite toward improving a robot's morphology. The algorithms presented in this chapter aim to iteratively update the morphological parameters of a robot (see §5.1) to avoid violating a set of limitation constraints imposed on the target system (see §5.1.2).

Robot Morphology For convenience, kinematics, geometric shape, and inertial configuration of a robot are referred to as its *morphology*, and the set of parameters for these attributes are referred to as the *morphological parameters*; this vector of parameters is denoted $\boldsymbol{p}$. Table 7 describes the morphological parameters considered in this dissertation. The morphological parameters included in $\boldsymbol{p}$ must be chosen with some care. The parametrization should be expressive enough to permit describing a wide range of morphologies, while remaining sufficiently restrictive to preclude any easily rejected morphologies.

5.1.1 Limitations on robot performance There is a volume of operational space—this dissertation expands the common definition of this term to encompass not just points in 3D, but twists and wrenches as well—that the robot must be able to access to achieve the desired aim of a given task (e.g., such as the task trajectory in Section 5.1.3). If a robot is unable to perform that task, then it must be the case that either some operational space states defined in the task are inaccessible to the robot due to some limitation acting on the physical aspects (geometric, kinematic, dynamic) of the system (e.g., maximum torque, kinematic reachability limits), or that the robot has entered an absorbing state in operational space (e.g., fallen over)—assuming the robot's control system is not capable of self-righting maneuvers. Task requirements indicate whether such failures have occurred and are defined qualitatively in each experiment.

A robot should be physically capable of performing a prescribed task once the trajectory and actuator forces required by a control policy to execute that task do not violate any limitations. In other words, *a task is feasible with respect to a robot once all states and actuator forces of the control policy satisfying the task lie within the region bounded by that robot's limitations*. A robot's limitations are represented as a set of algebraic constraints bounding a feasible region of generalized coordinates, generalized velocities, and actuator forces. Figure 28 illustrates a region of the actuator-constrained torque-speed space that a particular task might require.

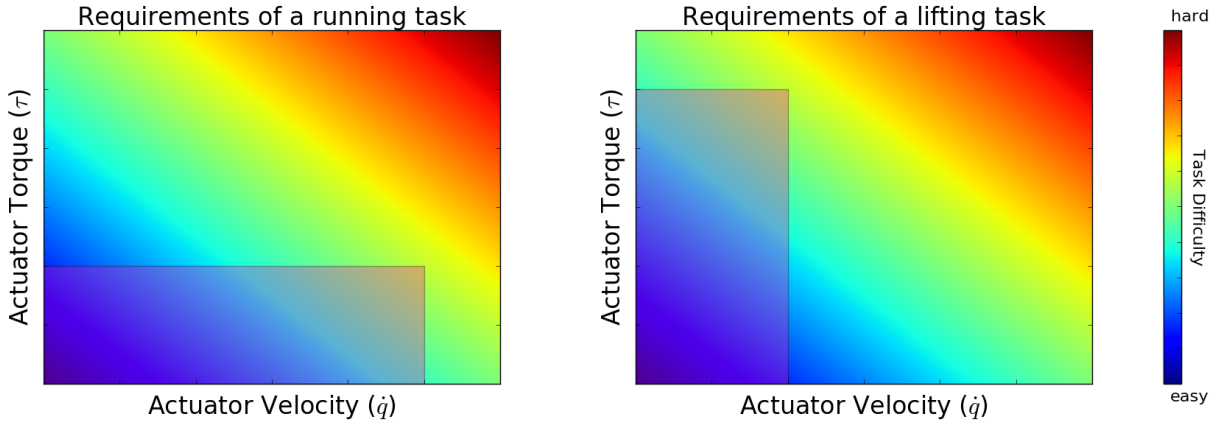


Figure 28: ***The requirements of a target task (shaded box) plotted on top of a scale of task difficulty (i.e., higher difficulty tasks require more power from robot actuators). A running task may necessitate a lower maximum torque τ and higher maximum joint velocities $\dot{q}$ than a task that involves lifting a heavy object. This plot should be compared against the explanations in Figure 30.***

A simple example of a robot limitation is the maximum torque output of an electromechanical actuator. Given such an actuator, there exists a speed beyond which torque output is zero, much like pedaling a fast bicycle in a low gear produces little effect. This mapping between torque output and actuator speed is typically referred to as the "torque-speed" curve (see e.g., Figure 29).

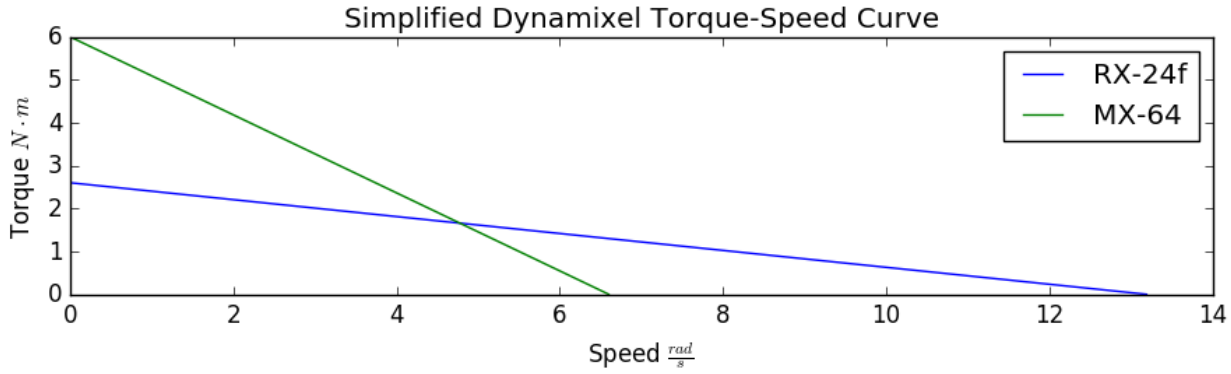


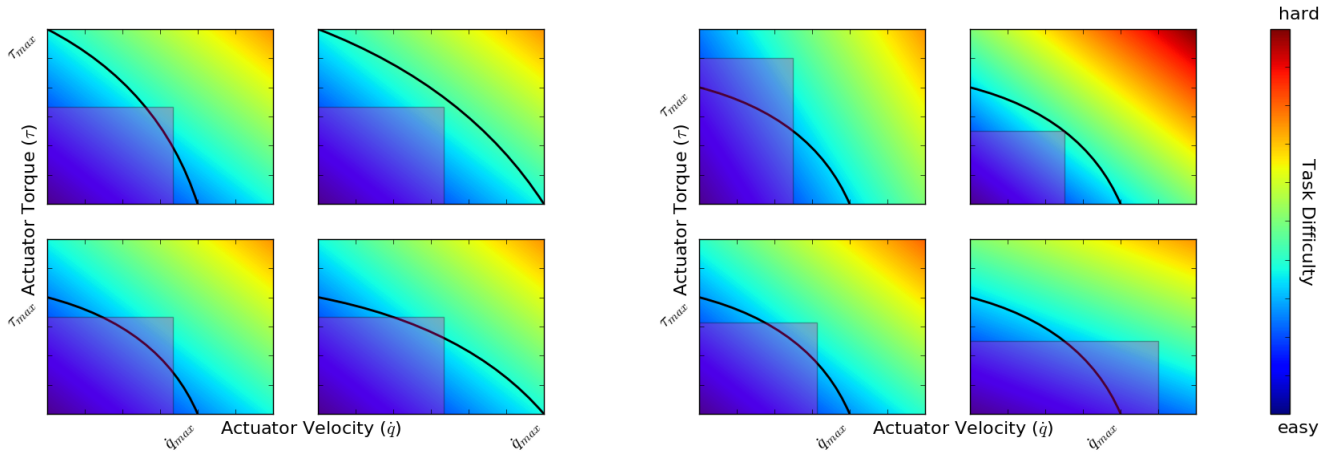
Figure 29: ***Torque-speed tradeoff for the* MX-64 *and* RX-24F *series Dynamixel actuators.***

Complexity of topological updates Adding joints or links to a robot's topological structure introduces complexities in planning and discontinuous jumps in witness function value. This dissertation considers only a fixed topology (one base with four, three-jointed limbs) and adjust the geometric, inertial, and actuation parameters of that model.

The robot modification approach presented in this chapter focuses on updating morphological

parameters to make performance of the tasks like robotic running feasible given fixed torque and speed limits: rather than consider swapping out actuators (with effects shown in Figure 30a), altering continuous parameters was considered (with effects shown in Figure 30b).

Given a fixed topological structure, there are two types of modeling parameters that are considered: continuous and discrete. This dissertation focused on updating continuous parameters; updating discrete parameters would allow modifying gear ratios and link fabrication materials, for example.



(a) *Actuator Modification*

Bottom Left: A robot whose actuators are not suitable to perform the target task will not fit the requirements of the task within its capabilities (under the torque-speed curve).

Top Left: Increasing the maximum torque of a robot's actuators τ_{max} enables the robot to achieve tasks that necessitate higher force output, but at the same actuator speeds as before the modification.

Bottom Right: Increasing the maximum actuator velocity $\dot{q}_{\text{max}}$ enables the robot to achieve tasks that necessitate higher speeds, but subject to the same actuator forces as before the modification.

Top Right: Increasing the maximum torque and maximum velocity enables the robot to perform all required force and speed requirements of the target task.

(b) *Morphological Parameter Modification*

Bottom Left: A robot whose actuators or morphology are not capable of performing the task will not be able to perform the motions required by the task.

Top Left: An increase in leg length, foot radius, or foot friction will result in a decrease in the actuator speed requirements for a task by providing the robot with a longer lever or greater reaction force when interacting with the environment.

Bottom Right: Decreasing leg length, foot radius, or foot friction will decrease the actuator torque requirements of a task by providing the robot a shorter lever or less reaction force when interacting with the environment.

Top Right: Adjusting the robot morphology in a directed manner will allow the robot to fully utilize its actuator capabilities, permitting it to avoid encountering a limitation while performing the motions required to realize a task.

Figure 30: ***The requirements of a target task (shaded box) sometimes lie outside of the a robot's capabilities, bounded in this plot by the torque-speed curve (under the curved line). Actuators (a) or morphological parameters (b) can be modified to increase the capabilities of the robot to fit a given task. If the robot's morphological parameters are updated carefully, the robot might become capable of performing a target task, even with a fixed torque-speed curve.***

5.1.2 Quantifying robot limitations for computer-aided design modification The algebraic functions representing mechanical limits in constrained rigid body dynamics (see definition of a DAE in Section 2.5.2) were found to often have a physical parallel that contributes to the dynamic stability of a robotic system—contact can restore balance and friction provides traction and application of contact force—permitting stabilization maneuvers such as catch-stepping and capture point stabilization (Raibert, 1986; Koolen et al., 2012); lack of contact provides freedom of motion for balancing behaviors such as the hip and angle strategy for humanoid balancing (Stephens & Atkeson, 2010b), as well as the ability to move to a capture point.

When a dynamical system switches modes (i.e., a constraint becomes either active or inactive), the simulated robotic system can experience a sudden, non-smooth jump in its state. Such a discontinuous jump in the dynamics might destabilize the robotic system, causing it to diverge from its expected behavior, sometimes catastrophically. This work attempts to correct for or mitigate the effect of such destabilizing behavior. A robot's design is modified to exhibit robust behavior by choosing models that minimally experience unanticipated mode switches.

Other limitations that are less apparent than the torque-speed trade-off may be equally important when gauging the capabilities of a robot. For example, a robot must avoid front-back foot collisions at high locomotion speeds when performing asymmetric gaits about the sagittal-plane (e.g., during walking and trotting). Additionally, certain robot designs may impose kinematic limitations on how high or far a robot can reach, preventing it from performing certain pick-and-place tasks, for example. A list of the robot limitations that have been considered in this work are listed in Table 4.

A robot's limitations are represented in this model modification approach using inequality constraints. Where the task requirements and objectives in Chapter 3 are boolean, *witness functions* provide a set of algebraic constraints bounding a feasible region of generalized coordinates, generalized velocities, and actuator forces. A witness function produces an m-dimensional vector of unit-less "distances" representing how close each constraint is to being exactly satisfied. The

Limits **Identified failure-inducing limits**	
Parameter (unit)	number of constraints
Roll limit of base (stability)	1
Pitch limit of base (stability)	1
Plan leaves kinematic reach of robot	1
Actuator torque & speed limits	$N_{\texttt{joints}}$
Actuator angle limits	$N_{\texttt{joints}}$
Self Collision (non-adjacent links)	$\binom{N_{\texttt{links}}}{2}$
Non-foot ground Collision	$N_{\texttt{links}} - N_{\texttt{feet}}$
Unexpected foot collision (scuff or stub)	$N_{\texttt{feet}}$

Table 4: *Limits to robotic performance considered in this work. Some of these examples may only apply to controlled or legged systems.*

witness functions depend on the robot's state $\{\boldsymbol{q}, \dot{\mathrm{q}}\}$ and input $\boldsymbol{u}$:

$$\Phi_{\boldsymbol{p}_j}(\boldsymbol{q}, \dot{\mathrm{q}}, \boldsymbol{u}) \geq 0 \tag{20}$$

A witness function $\boldsymbol{\Phi_p} : \mathbb{R}^{n_q \times n_v \times n_u} \rightarrow \mathbb{R}^m$ (where n_q is the dimension of robot generalized coordinates, n_v is the dimension of robot generalized velocities, and n_u is the dimension of the *controllable* generalized forces) informs us whether on of these limitations has been reached or exceeded. Auxiliary variables are sometimes added to a simulated system to make building good witness functions that does not change rapidly over a small time interval easier, which would reduce the likelihood of missed limit violations.

If there exists an index j such that constraint $\Phi_{\boldsymbol{p}_j} < 0$, constraint j has been violated. Negative witness function values indicate by how far a limit has been exceeded, while positive values represent distance from the limit boundary. The model modification approach presented in this chapter aims to transform the generalized coordinates, velocities, and actuator forces *or the definitions of these spaces* of a robot such that the new coordinates, etc. lie within their respective feasible regions while the robot performs a given task.

5.1.3 Control Policy: task trajectory The present focus of updating a robot's morphology leads to a preference that the operational space trajectories produced by a robot's control policy remain constant as the robot's shape is adjusted; the robot's feet or hands should move along the same

trajectory in the world (i.e., aim at the same intractable objects or terrain features), even though its configuration space trajectory might change as morphological parameters are updated. The task trajectory $\mathbf{T}$ is a vector of *operational space* states for key links of the robot (as defined by the user).

$$\mathbf{T} \equiv \begin{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \\ \aleph_1 \end{bmatrix}^\mathsf{T} & \cdots & \begin{bmatrix} \boldsymbol{x}_i \\ \\ \aleph_i \end{bmatrix}^\mathsf{T} & \cdots & \begin{bmatrix} \boldsymbol{x}_n \\ \\ \aleph_n \end{bmatrix}^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{21}$$

The vector of states is sampled from a locomotion control policy (Chapter 9) at the control loop frequency of the target physical robot's control system. For a locomoting quadrupedal robot, these key links might be the base and each foot:

$$\boldsymbol{x} \equiv \left[\boldsymbol{x}_{\text{base}}^\mathsf{T}, \boldsymbol{x}_{\text{LF foot}}^\mathsf{T}, \boldsymbol{x}_{\text{RF foot}}^\mathsf{T}, \boldsymbol{x}_{\text{LH foot}}^\mathsf{T}, \boldsymbol{x}_{\text{RH foot}}^\mathsf{T}\right]^\mathsf{T} \tag{22}$$

$$\aleph \equiv \left[\aleph_{\text{base}}^\mathsf{T}, \aleph_{\text{LF foot}}^\mathsf{T}, \aleph_{\text{RF foot}}^\mathsf{T}, \aleph_{\text{LH foot}}^\mathsf{T}, \aleph_{\text{RH foot}}^\mathsf{T}\right]^\mathsf{T} \tag{23}$$

Where $\boldsymbol{x}_{\text{base}} \in \text{SE}(3)$, $\aleph_{\text{base}} \in \mathbb{R}^6$, $\boldsymbol{x}_{\text{foot}} \in \mathbb{R}^3$, and $\aleph_{\text{foot}} \in \mathbb{R}^3$ for a floating-base quadrupedal robot with three actuated joints on each leg. In order to follow an assigned task trajectory, the robot must convert the operational space state $[\boldsymbol{x}_i^\mathsf{T}, \dot{\aleph}_i^\mathsf{T}]^\mathsf{T}$ at discrete sample i in the task trajectory $\mathbf{T}$ (with time $\Delta t \ll 1$ between successive samples) to generalized state $[\boldsymbol{q}_i^\mathsf{T}, \dot{\mathbb{q}}_i^\mathsf{T}]^\mathsf{T}$ and a set of actuator forces $\boldsymbol{u}_i$ to reach the next generalized velocity $\dot{\mathbb{q}}_{i+1}$ defined by the spatial link velocities $\dot{\aleph}_{i+1}$. Generic inverse kinematics (IK) and inverse dynamics (ID) functions are defined to represent these calculations with respect to a robot parameters $\boldsymbol{p}$:

$$\{\boldsymbol{q}_i, \dot{\mathbb{q}}_i\} = \text{INVERSEKINEMATICS}_{\boldsymbol{p}}(\boldsymbol{x}_i, \dot{\aleph}_i) \tag{24}$$

$$\{\boldsymbol{u}_i\} = \text{INVERSEDYNAMICS}_{\boldsymbol{p}}(\boldsymbol{q}_i, \dot{\mathbb{q}}_i, \dot{\boldsymbol{q}}_{i+1}, \Delta t) \tag{25}$$

5.2 Iterative robot design

Figure 31 depicts a flow chart of the model modification process as the robot iterates through the task trajectory; the following steps are taken:

1. The IK problem is solved for the current operational space configuration and spatial velocity (i.e., at discrete sample i) and the next operational space configuration and spatial velocity (i.e., at discrete sample $i + 1$).

2. Actuator torques ($\boldsymbol{u}_i$) are computed that will yield the next (i.e., at discrete sample $i+1$) desired generalized velocity using ID and the configuration space state and desired generalized velocity at iteration i.

3. The robot applies actuator torques $\boldsymbol{u}_i$ to the robot yielding generalized state $\{\boldsymbol{q}', \dot{\mathfrak{q}}'\}$, with the expectation that the discretization is sufficiently small such that $\boldsymbol{q} \approx \boldsymbol{q}'$ and $\dot{\mathfrak{q}} \approx \dot{\mathfrak{q}}'$.

4. The witness functions are evaluated against actuator commands and desired configuration space state (at sample i) and desired next state (for $i + 1$).

5. If a witness function becomes negative during the robot's virtual performance of the task from sample i to sample $i + 1$, the model is updated to correct the limit violation.

For example, if an actuator torque limit violation is observed at sample i ($u_i^j > u^{j^{\max}}$), the robot's morphological properties can be updated to produce a lower magnitude u_i^j toward realizing the operational space trajectory (by, e.g., reducing leg length, decreasing foot friction, or reducing the weight parameterization for the joint's child links). Algorithm 2 incrementally pushes the parameterization toward the feasible region.

Finally, noting a few subtle aspects of the "controller" in Figure 31: (1) the controller need only handle small disturbances because the state is reset to $\{\boldsymbol{q}_i, \dot{\mathfrak{q}}_i\}$ on each iteration; (2) no gain tuning is necessary; and (3) the ID controller can assess the peak performance of the robot without specific knowledge of the servo-motor control loop gains on actual hardware, while the error-feedback controller used *in situ* on the robot is expected to track trajectories with some error.

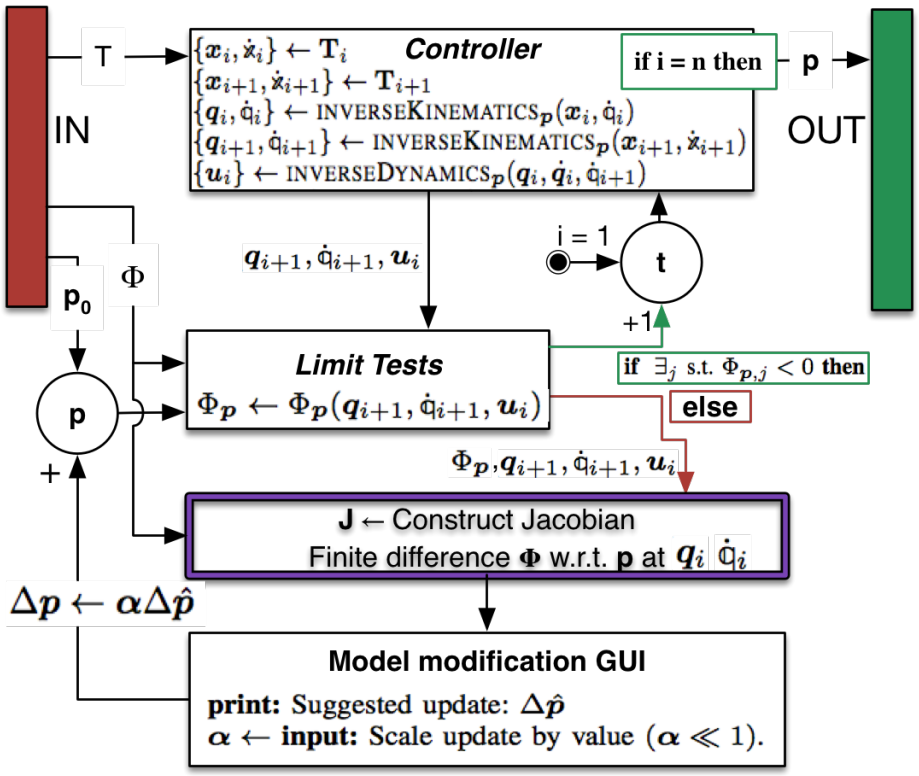


Figure 31: ***A flowchart visualization of Algorithm 2:*** UPDATEMORPHOLOGY(.) ***with input parameter*** width ***set to a value of 1 (no dynamic simulation). The algorithm takes an operational space trajectory*** ($\mathbf{T}$) ***as input and updates the robot's morphological parameters*** (p) ***until the set of witness functions all evaluate to non-negative limit values (i.e.,*** $\Phi_p(q, \dot{q}, u) \geq 0$***) during operation. The "Construct Jacobian" block represents a condensed version of the flowchart in Figure 33.***

Difficulty scaling Constraint violation might be large immediately upon executing a difficult task with a novel robot design. For example, if the actuators of a walking robot are very weak (i.e., can supply very little torque before reaching saturation), large torque limit violations might immediately be observed for this robot when attempting a running gait.

To address this problem, the user can bootstrap the robot modification process by progressively scaling the task "difficulty". For a pick-and-place task, this may entail starting with an easier task such as lifting a lightweight object or performing slower movements; such bootstrapping is straightforward for locomotion tasks where lower velocity gaits are typically easier to perform. Once the robot can successfully complete an easier version of the task, the user would repeat the process on progressively harder versions of the task.

Algorithm 2 $\{\boldsymbol{p}\} = \text{UPDATEMORPHOLOGY}(\boldsymbol{p}_0, \mathbf{T}, \boldsymbol{\Phi}, \text{width})$ Starting at initial morphological parameters $\boldsymbol{p}_0$, updates $\boldsymbol{p}$ to keep all elements of $\boldsymbol{\Phi}(\boldsymbol{q}, \dot{\mathbb{q}}, \boldsymbol{u})$ non-negative while following operational space task trajectory $\mathbf{T}_{1..N}$. The algorithm resets the state of the robot to the desired state after simulating the system over `width` discrete increments (i.e., a sliding window) of the task trajectory. Returns the new set of morphological parameters $\boldsymbol{p}$.

$\boldsymbol{p} \leftarrow \boldsymbol{p}_0$
$i \leftarrow 0$
do ▷ (see §5.2)
  $\{\boldsymbol{x}_i, \dot{\mathbb{x}}_i\} \leftarrow \mathbf{T}_i$
  $\{\boldsymbol{q}_i, \dot{\mathbb{q}}_i\} \leftarrow \text{INVERSEKINEMATICS}_{\boldsymbol{p}}(\boldsymbol{x}_i, \dot{\mathbb{x}}_i)$
  $j \leftarrow i$
  do ▷ Sliding window inner loop §5.2.1
    $\{\boldsymbol{x}_{j+1}, \dot{\mathbb{x}}_{j+1}\} \leftarrow \mathbf{T}_{j+1}$
    $\{\boldsymbol{q}^{\text{des}}_{j+1}, \dot{\mathbb{q}}^{\text{des}}_{j+1}\} \leftarrow \text{INVERSEKINEMATICS}_{\boldsymbol{p}}(\boldsymbol{x}_{j+1}, \dot{\mathbb{x}}_{j+1})$
    $\{\boldsymbol{u}_j\} \leftarrow \text{INVERSEDYNAMICS}_{\boldsymbol{p}}(\boldsymbol{q}_j, \dot{\mathbb{q}}_j, \dot{\mathbb{q}}^{\text{des}}_{j+1}, \Delta t)$
    $\{\boldsymbol{q}_{j+1}, \dot{\mathbb{q}}_{j+1}, -\} \leftarrow \text{SIMULATOR}_{\boldsymbol{p}}(\boldsymbol{q}_j, \dot{\mathbb{q}}_j, \boldsymbol{u}_j)$ ▷ Integrate physical simulation (Equation 17)
    $\boldsymbol{\ell}_{\boldsymbol{p}} \leftarrow \boldsymbol{\Phi}_{\boldsymbol{p}}(\boldsymbol{q}_{j+1}, \dot{\mathbb{q}}_{j+1}, \boldsymbol{u}_j)$ ▷ Evaluate witness functions
    if $\exists_k$ s.t. $\boldsymbol{\ell}_{\boldsymbol{p},k} < 0$ **then**
      $\mathbf{J} \leftarrow$ Construct Jacobian (see §5.2.2)
      $\Delta\hat{\boldsymbol{p}} \leftarrow -\mathbf{J}^{\dagger}\boldsymbol{\ell}_{\boldsymbol{p}}$
      print: Suggested update: $\Delta\hat{\boldsymbol{p}}$
      $\boldsymbol{\alpha} \leftarrow$ **input:** Scale update by value ($\alpha \ll 1$).
      $\Delta\boldsymbol{p} \leftarrow \alpha\Delta\hat{\boldsymbol{p}}$
      $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta\boldsymbol{p}$ ▷ Update robot parametrization.
      $j \leftarrow i$ ▷ Reset current window
    else
      $j \leftarrow j + 1$
  while $j < i + \text{width}$
  $i \leftarrow i + 1$
while $i < N$
return $\{\boldsymbol{p}\}$ ▷ Successfully updated robot to perform task $\mathbf{T}$.

Algorithmic correctness There do not exist tractable algorithms for determining whether one or more nonlinear inequality constraints (i.e., limit functions) are satisfiable over a discretization of a continuous trajectory; this problem generally corresponds to a non-convex optimization problem. In the absence of a tractable approach to an optimal solution, several alternatives are considered, including optimizing parameters over the entire execution of a trajectory (somewhat similar to trajectory optimization) and optimizing parameters over a "window" sliding over the trajectory. The latter approach was explored in this work, since it was simpler, with the understanding that this approach was liable to modify the parameters one way at time t_a during the trajectory execution only to revert this change at time $t_b > t_a$. This phenomenon was not encountered when testing the model modification approach but note that the user supervision would be able to prevent such regressions.

Control system (for improvement algorithm) The algorithm in Figure 31 checks each portion of the trajectory for validity by generating desired configuration space states and commands from a provided operational space trajectory $\mathbf{T}$. The robot is initialized at the desired state from $\mathbf{T}_i$, then controlled to reach the subsequent desired state at $\mathbf{T}_{i+1}$. The algorithm uses a closed loop error-feedback controller that applies feedback desired velocity updates to an inverse dynamics controller with contact force prediction (see e.g., Righetti et al. (2013)) to instantaneously (i.e., in one control loop iteration) correct for trajectory tracking error using actuator forces.

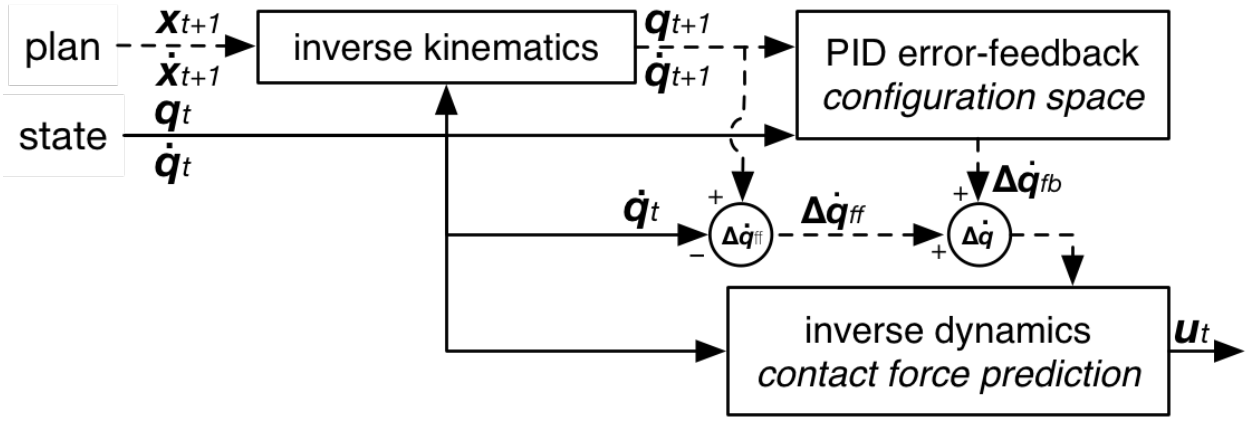


Figure 32: ***Control system used by the iterative robot modification and testing algorithm. Stabilization and error-feedback are accumulated as velocity updates and input into inverse dynamics controller. An inverse dynamics controller is used to determine the actuator torques u and contact forces acting on the robot, subject to the state ($q_i, \dot{q}_i$), and contact configuration at sample i.***

5.2.1 Considering coordinate based witness functions with a "sliding window" The flowchart in Figure 31 depicts a flowchart visualization of Algorithm 2 with its input parameter width set to a value of 1, corresponding to no dynamic simulation; this means that input forces and goal state are checked against the limit functions and then the robotic system is reset to the next desired state. In order to consider coordinate and stability based witness functions (e.g., unwanted contact, the ZMP leaving the stable region within the contact polygon, missed contact, and others listed in Table 4), drift in the robot's state away from plan must be considered (due to e.g., external perturbations, poor model fidelity, or bad control input). The sliding window inner loop of Algorithm 2 allows the dynamical system to evolve according to its control policy (desired trajectory in this case) for a duration of time; allowing such divergences to take effect permits the detection (and mitigation of) coordinate based limit violations. The updating process demonstrated in Chapter 6 only requires that $\mathrm{width} = 1$ because desired velocity and input torque are not coordinate based limitations and can be checked instantaneously.

5.2.2 Relating witness functions to design and controller parameters Modifying the morphological parameters affects when and whether the limits in Table 5 become violated when executing a desired trajectory. Each model update of the robot modification process requires constructing

a gradient for how each model parameter affects each active limit function at the moment in the task where a limit function first becomes active. Gradient vector $\nabla_j \Phi_{\texttt{model}(i)}(\boldsymbol{q}, \dot{\mathrm{q}}, \boldsymbol{u})$ is computed using finite differencing (as described in Figure 33) with respect to modeling parameter j of m total parameters.

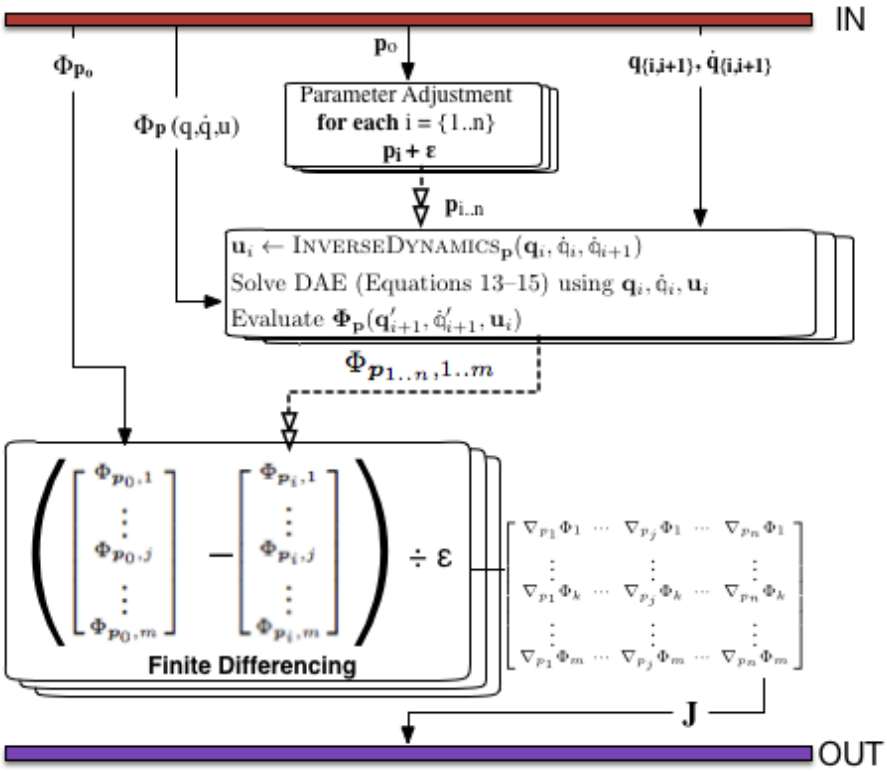


Figure 33: ***Jacobian generation flowchart.***

The limit gradients are concatenated to form a limit Jacobian, $\mathbf{J}_\Phi$, that describes the slope of the constraint-parameter space at a specific point in state space $(\boldsymbol{q}, \dot{\mathrm{q}})$ under the current model parameters $(\boldsymbol{p}_i)$.

$$\mathbf{J} = \left[\nabla_0 \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\mathrm{q}}, \boldsymbol{u}) \cdots \nabla_m \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\mathrm{q}}, \boldsymbol{u})\right]$$

The gradient descent direction $-\mathbf{J}^\dagger \boldsymbol{\ell}$, where † is the pseudo inversion operator, yields an update direction to the robot modeling parameters that should decrease the limit function violations ($\boldsymbol{\ell} = \Phi_{\boldsymbol{p}_i}(\boldsymbol{q}, \dot{\mathrm{q}}, \boldsymbol{u})$). If $\mathbf{J}$ defines an under-determined system (i.e., there are many parameter updates

that produce the same effect on the active witness functions and $\mathbf{J}^\top\mathbf{J}$ is not invertible) $\boldsymbol{z} = \mathbf{J}^\dagger\boldsymbol{b}$ will evaluate to the minimum Euclidian norm solution of $\mathbf{J}\boldsymbol{x} = \boldsymbol{b}$ (i.e., will produce the minimum model parameter update to correct the limit violation). The current modeling parameters ($\boldsymbol{p}_i$) can then be updated to those at the next iteration $\boldsymbol{p}_{i+1}$:

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i - \alpha\mathbf{J}^\dagger\boldsymbol{\ell}$$

for $\alpha \ll 1$ (α is tuned manually). The designer incorporates expert knowledge into this tuning stage, weighting the gradient update and perhaps making additional modifications to the robot model. Specifically, a designer is permitted to roll-back an update or apply either of two changes repeatedly to the update: (1) zeroing any element of the parameter update vector; (2) applying front-hind symmetry to a specific parameter's update.

5.3 Conclusion

The notion of "performance" is often challenging to condense to a single scalar value but hard limits (like joint range-of-motion limits, torque limits, and contact constraints) are straightforward to specify. This chapter used that idea and a human designer's intuition and analysis to guide a robot's hardware and software toward better performance; computational tools can keep those designs inside the space that the hardware is capable of working effectively within. Whether robustness is dominated by a control policy or a robot's design is uncertain; the systems are coupled and the robustness of one system can not be verified without the presence of the other. The following chapter, Chapter 6 will evaluate the performance of the model modification process from this chapter on low-cost quadrupedal robots, with the goal of helping them walk faster and exhibit fewer failures; it demonstrates how the morphological parameters for various locomoting quadrupedal robots can be adjusted with the presented robot modification process to improve performance *in situ*.

6 Virtual Prototyping: simulation-assisted robot design

When robots engage in highly dynamic activities like running, performance often bumps against one or more geometric, kinematic, or dynamic constraints. Electromagnetic actuator torque-speed curves typify such a tradeoff: enough actuator torque is required to accelerate a legged robot to a run, but gearing a robot to a higher gear ratio to produce more torque reduces maximum actuator velocity, and by extension, maximum forward speed.

The particle traces approach described in Chapter 3 helps detect failures in a control policy that result from robot modeling error, state uncertainty, control delay, etc. (i.e., infidelity in the virtual representation of a real system). The design improvement tool presented in Chapter 5 assesses the performance of a robot on a task and then updates the robot's morphological parameters to avoid dangerous regions of its state and control space. This chapter uses the robot modification algorithm from Chapter 5 to analyze the trajectory that a robot would execute repeatedly in order to perform a task (quadrupedal locomotion, in the target application). As the robot virtually executes an operational space trajectory the robot's control system determines the motor torques and generalized configurations and velocities necessary to move between each successive pair of points in operational space. If one of these iterations violates one of the limits set on the robot, the model modification software updates the robot's morphological parameters to correct for the limit violation.

Two example applications for the model modification approach are presented in this chapter to illustrate the effectiveness of the approach; Section 6.1 presents an overview of the experimental setup and procedure common to both testing scenarios. The first example demonstrates the design of a 3D printed quadruped that is too weak and delicate to locomote with an arbitrarily assigned morphological design (§6.2); the second example demonstrates the redesign of a physically reconfigurable quadruped to improve its initially poor locomotion performance (§6.3). The chapter concludes in Section 6.4 with a discussion about how the design improvements made to robots in this chapter can improve robot robustness and be used to mitigate the various divergent

behaviors detected by the particle traces and control policy falsification procedures presented in this dissertation.

6.1 Overview of virtual and *in situ* tests

Focusing on torque-speed curves as just one application, this chapter demonstrates how the model modification process is able to adjust a legged robot's parameterization from a point on the outside of the feasible torque-speed curve to within the feasible torque-speed region, all without compromising locomotion performance.

6.1.1 Robot Limitations Actuator torque-speed curves are used in this chapter as one commonly encountered example of the "limitations" referred to in Section 5.1.2. Both assessed robots were assembled from inexpensive hardware fabricated using simple manufacturing techniques; each robot utilizes twelve actuators with independently-defined torque-speed constraints (see Table 5). One of the robots (§6.3) uses two types of actuators, giving different joints different actuation limitations.

`Limits`
Identified failure-inducing limits

Parameter (unit)	number of constraints
Actuator torque & speed limits	$N_{\texttt{joints}}$
Actuator angle limits	$N_{\texttt{joints}}$

Table 5: *Limits to robotic performance considered when improving the robot.*

The proximity that each of the robot's actuators is to violating each of these actuator limitations is tracked by a piecewise witness function Φ_i for each actuator $i \in \{1, ..., 12\}$:

$$\Phi_i = \begin{cases} \tau_i^{\max} - v_i(\tau_i^{\max}/v_i^{\max}) - \tau_i \geq 0, & \text{if} \quad v_i \geq 0,\ \tau_i \geq 0 \\ \tau_i^{\max} + v_i(\tau_i^{\max}/v_i^{\max}) - \tau_i \geq 0, & \text{if} \quad v_i < 0,\ \tau_i \geq 0 \\ -\tau_i^{\max} - v_i(\tau_i^{\max}/v_i^{\max}) + \tau_i \geq 0, & \text{if} \quad v_i \geq 0,\ \tau_i < 0 \\ -\tau_i^{\max} + v_i(\tau_i^{\max}/v_i^{\max}) + \tau_i \geq 0, & \text{if} \quad v_i < 0,\ \tau_i < 0 \end{cases}$$

The witness function constrains a volume of the torque-speed space and includes the origin; it

returns a non-negative value for all torque and speed values between each curve segment and the origin. Figure 45 shows an example of the torque-speed limit witness function plotted with the torque-speed curve. The positive quadrant of this torque-speed curve constraint is illustrated in Figure 34. Figures 38 and 44 specify which of these constraints is used in each experiment.

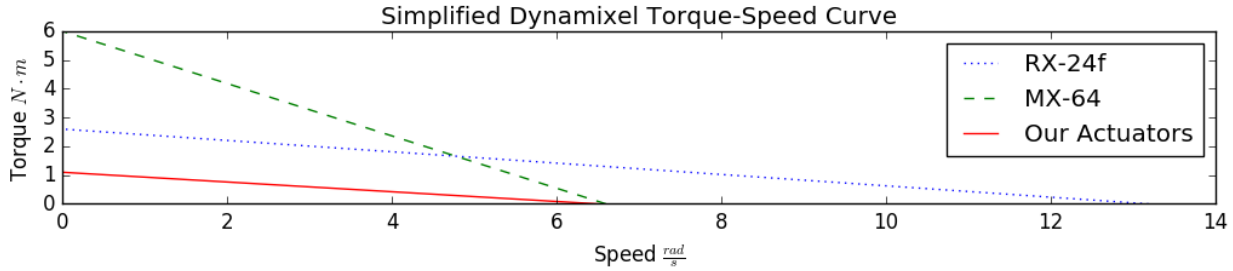


Figure 34: ***A plot of a linear torque-speed curve for the* MX-64 *and* RX-24F *series Dynamixel actuators are compared against the hobby servos used in this experiment.***

Gait The following two validation sections use a locomotion control policy that is best suited to their scale; the control parameters are described in their respective experimental sections. The gait planner generates desired foot positions and velocities based on the current and desired velocity of the base (see Figure 66).

Controller The control diagram for experimentation is illustrated in Figure 35. Unlike the "controller" described in §5.2, the simulated robot is controlled by an error feedback control scheme, and the *in situ* robot is controlled open-loop, by sending desired positions to the DYNAMIXEL actuators.

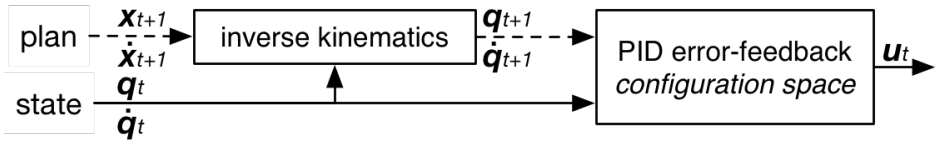


Figure 35: ***Control system used by the robot* in situ *for validating the robot morphological improvement process.***

Quadrupedal Robot The following two validation sections use differently sized quadrupedal robots with topological structures identical to the previous quadrupeds described in Chapter 4. All are 18 degree-of-freedom (12 actuated) quadruped robots; link names referred to throughout this

chapter are labeled in Figure 36.

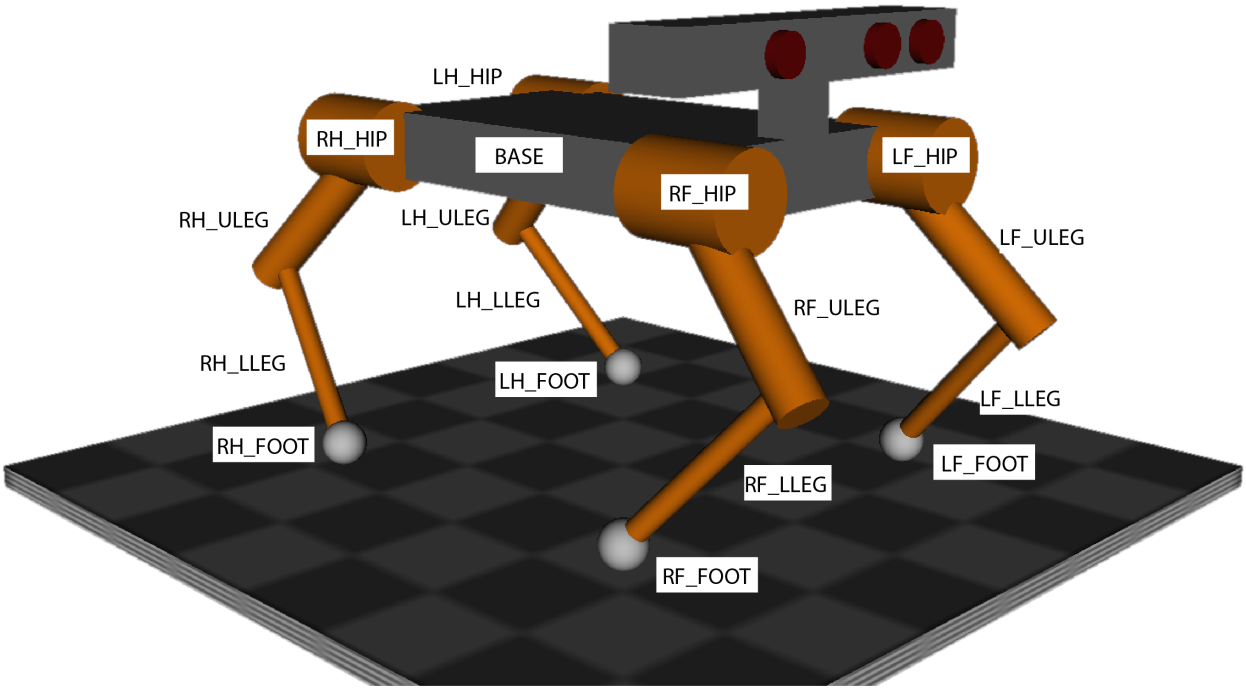


Figure 36: ***A virtual rendering of the simulated reconfigurable robot; its geometric, kinematic, and inertial models closely match the physical robots in this section despite differences in appearance. The base link (grey) has a box geometry; limbs, originating from the corners of the base have a cylindrical geometry; feet have a spherical geometry. All links also have a density that, with the calculated volume, determine the mass of the link. Densities were selected based on the building material used for the robot (e.g., the 3D printed robot has a maximum link density of PVC and could be printed at lower densities by hollowing out the link). An RGBD sensor on the "head" (unused) is rendered also.***

6.2 Designing a 3D printed Robot

The interactive design approach is tested in this section by generating robot designs from both "supervised" (human in-the-loop) and fully automated processes and then compare their relative performance with respect to a simple locomotion task. Experiments with two fabricated robots are described in this section; the robots were 3D printed using morphological and mass parameters of the tuned model. The human-in-the-loop was permitted to perform only simple adjustments to each automatically-generated model update (see §5.2.2).

A test of the robot design framework was first performed and then an experiment was run to assess the validity of the simulation results against 3D printed robots. The evaluated task was focused on locomoting in a straight-line across a planar environment, though the approach is not limited to

this scenario. The objective was to update the morphological parameters of a quadrupedal robot to perform a trot at a variable forward velocity. This experiment sought to adjust the continuous geometric and mass parameters of the links of the quadruped in order to achieve the highest speed possible given known actuator stall torque and velocity limits. The model modification process from Chapter 5 was used to iteratively update the robot model in order to increase its maximum trotting speed. Trotting was the targeted task in this chapter because it is a highly dynamic, known challenging task that requires a robot to make full use of its physical ability. After verification in simulation, changes to the virtual robot model were manually applied to the physical robot; observations focused on whether the improvements made in simulation also resulted in better *in situ* performance.

6.2.1 Experimental design To determine the usefulness of the proposed interactive design process, three candidate robot models were compared: (1) an **`initial`** robot, taken without modification from previous, successful locomotion experiments (see Chapter 4); (2) an updated robot design created by following an **`automated`** approach, seeded from the `initial` robot and allowing the design software to iteratively update the model according to the path of steepest descent until the model modification process could make no further progress; and (3) an updated robot design created by following a **`supervised`** approach, seeded from the `initial` robot using the software to guide the designer through modifications using the direction of steepest descent. After generating the three candidate models, the `initial` and `supervised` robot models were fabricated for testing *in situ*. The `automated` model was not fabricated, as its design was not limited to adhere to the 3D printer's fabrication constraints; I comment on this model in Section 6.2.4.

Both fabricated robots were tested by comparing their average velocity over a duration of trotting against the intended trotting speed. This metric was used to determine whether *in situ* performance improved at each target gait velocity for the `supervised` robot over the `initial` robot. It was expected that the `supervised` robot would be much more successful at performing the higher speed gaits, as it was modified to respect its actuator limitations for those gaits.

The updated robot designs (`automated` and `supervised`) were seeded from a simple robot

model (`initial`). A CAD model of the `initial` morphology designed for a 3D printer is depicted in Figure 43.

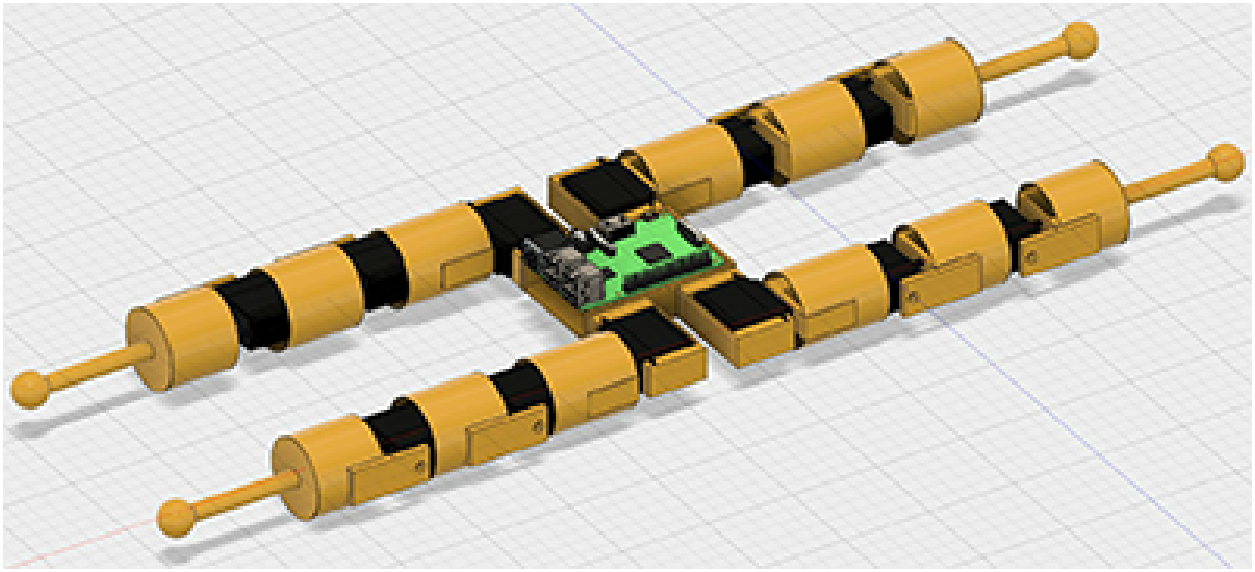


Figure 37: ***The `initial` robot design used to seed the updated models.***

6.2.2 Morphological parameterization One simplification made for the scenario in this section assumes that a locomoting robot will need to be symmetric across the sagittal plane. Because any robots that are laterally asymmetric are likely to be rejected by a designer, only the model parameters of half of the robot were adjusted, then those parameters were reflected to the other side when the model was updated.

Relevant morphological parameter and actuator limitation for this scenario, a 3D-printed quadrupedal robot, are shown in Figure 38.

Update Morphology: *3D-printed, low-cost, quadruped*

With morphological parameters $\boldsymbol{p}$:

Morphological Parameter (unit)	Links Affected
front, upper limb length, $l_{\text{Front,ULeg}}$ (cm)	{LF, RF}_{ULEG}
front, lower limb length, $l_{\text{Front,LLeg}}$ (cm)	{LF, RF}_{LLEG}
hind, upper limb length, $l_{\text{Hind,ULeg}}$ (cm)	{LH, RH}_{ULEG}
hind, lower limb length, $l_{\text{Hind,LLeg}}$ (cm)	{LH, RH}_{LLEG}
base length, l_{Base} (cm)	BASE
base width, w_{Base}(cm)	BASE
base height, h_{Base}(cm)	BASE
front foot radius, $r_{\text{Front,Foot}}$ (cm)	{LF, RF}_FOOT
hind foot radius, $r_{\text{Hind,Foot}}$ (cm)	{LH, RH}_FOOT
front limb radius, $r_{\text{Front,Limb}}$	{LF, RF}_{HIP, ULEG,LLEG}
hind limb radius, $r_{\text{Hind,Limb}}$	{LH, RH}_{HIP, ULEG,LLEG}
front, hip mass, $m_{\text{Front,Hip}}$ (cm)	{LF, RF}_{HIP}
front, upper limb mass, $m_{\text{Front,ULeg}}$ (cm)	{LF, RF}_{ULEG}
front, lower limb mass, $m_{\text{Front,LLeg}}$ (cm)	{LF, RF}_{LLEG, FOOT}
hind, hip mass, $m_{\text{Hind,Hip}}$ (cm)	{LH, RH}_{HIP}
hind, upper limb mass, $m_{\text{Hind,ULeg}}$ (cm)	{LH, RH}_{ULEG}
hind, lower limb mass, $m_{\text{Hind,LLeg}}$ (cm)	{LH, RH}_{LLEG, FOOT}

Such that *limitations* are not exceeded

Parameter	number of actuators	Value
Actuator Torque Limits	12	1.1 N·m
Actuator Velocity Limits	12	6.55 rad/s

Figure 38: ***Parameters and limits for the 3D-printed, low-cost, quadruped scenario. Limits to robotic hardware considered in the experiment are the stall torque and max velocity of a cheap hobby servo. A plot of a linear torque-speed curve described by these values is drawn Figure 34.***

6.2.3 Gait Parameterization Planning and control for the 3D printed robots were performed by a Raspberry Pi computer looping over a prerecorded configuration space trajectory for the specific robot morphology. The trotting gait trajectory was generated using the PACER planning and control software (Zapolsky, 2015). Gait parameters are defined in Figure 6.

***6.2.4 Robot performance* in sim** This section presents *in sim* results from the morphological modification process. Both `supervised` and `automated` design processes began using the `initial` model attempting to perform a trotting gait (see Table 6) at the starting forward velocity

Quadrupedal Gait Parameters

Parameter (unit)	Value
forward velocity (cm/s)	50
base linear offset (cm)	$\{0, 0, 0.75 \times$ min-leg-length$^*\}$
base orientation offset (rad)	$\{0, 0, 0\}$
step height (cm)	1
stance length (% base length)	200
stance width (% base width)	110
gait duration (sec)	0.3
liftoff timing (% gait duration)	$\{25, 75, 25, 75\}$
duty factor (% gait duration)	$\{60, 60, 60, 60\}$

Table 6: *Gait parameters for the trotting task performed by the robot. (*)*min-leg-length *refers to the minimum value of* $l_{\text{Front,ULeg}} + l_{\text{Front,LLeg}}$ *and* $l_{\text{Hind,ULeg}} + l_{\text{Hind,LLeg}}$ *between front and hind leg pairs.*

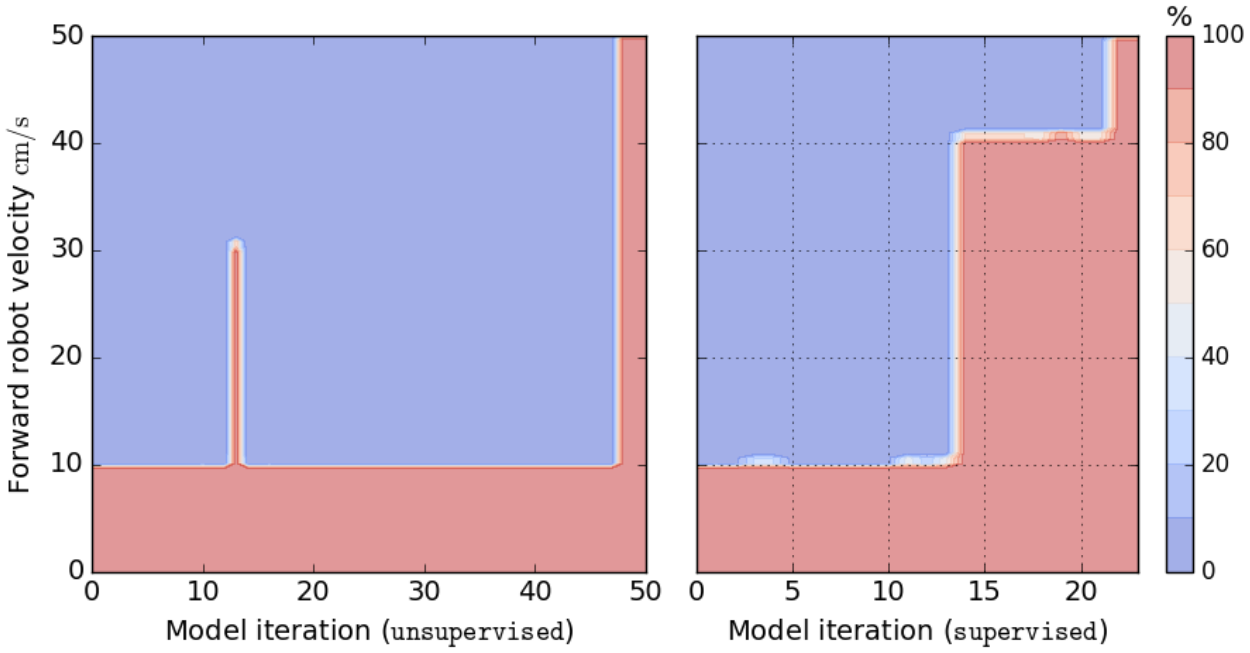


Figure 39: ***The `automated` (left) and `supervised` (right) robot design progress over several virtual model updates. Regions of the plot are colored according to the percentage of the gait that the robot model can complete at the specified velocity before violating an actuator limit. Colors blue, white, and red coincide with 0, 50, and 100 percent task completion, respectively.***

of 10 cm/s.

The result of the **human in the loop modification process** generated the morphology depicted (as a CAD model) in Figure 40. The **automated process** generated the morphology designed for a 3D printer depicted in Figure 41. The `automated` model yielded a robot characterized by extreme values (e.g., 1 g front foot mass and 187 mm hind foot radius). Although fabrication of this robot is possible, hard to quantify issues—such as self collision and thin, brittle link geometries—likely would have resulted in the robot destroying itself during the first test. Section 6.2.5 will note that the seemingly more robust `initial` model broke in the fifth trial during physically situated testing.

Figure 39 shows that modifying the robot enough to progress past one limit violation was usually enough to progress through the remainder of the planned gait, at least for a walking trot. This phenomenon expresses itself in the plots as either 0% or 100% progress in the plot (blue or red, respectively), and there are very few examples of 50% progress (white). *Both designs achieved the same maximum forward trotting velocity of 50 cm/s in simulation*, leading to the conclusion that the automated process resulted in a objectively equivalent, yet qualitatively inferior robot (as will be demonstrated shortly §6.2.5).

***6.2.5 Results* in situ** This section presents *in situ* results from the morphological modification process for the 3D printed robot. Both `supervised` and `automated` design processes began using the `initial` model attempting to perform a trotting gait at a starting forward velocity of 10 cm/s. The model needed to be improved to proceed to the next velocity (velocities were increased in 10 cm/s increments)

The situated robot was controlled at a forward velocity of 16.26 cm/s, determined by the approximately 60 Hz publish rate of the Raspberry Pi computer: attempting faster movement produced jumpy behavior when the robot tried to follow the joint trajectory in real-time.

The `initial` robot trotted an average distance of 64 cm over 20 gait cycles, or about 12.22 seconds of trotting during the trials where it managed to complete the task. The robot successfully

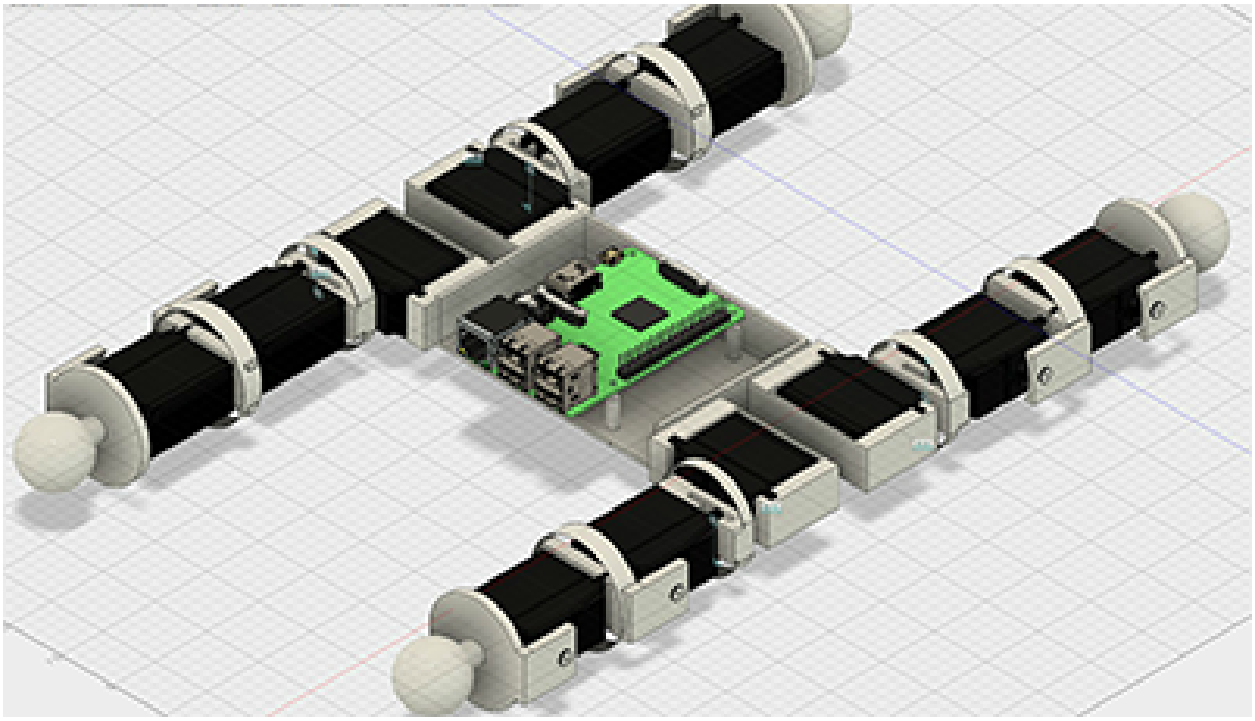


Figure 40: ***The `supervised` modified robot design produced using the interactive design process.***

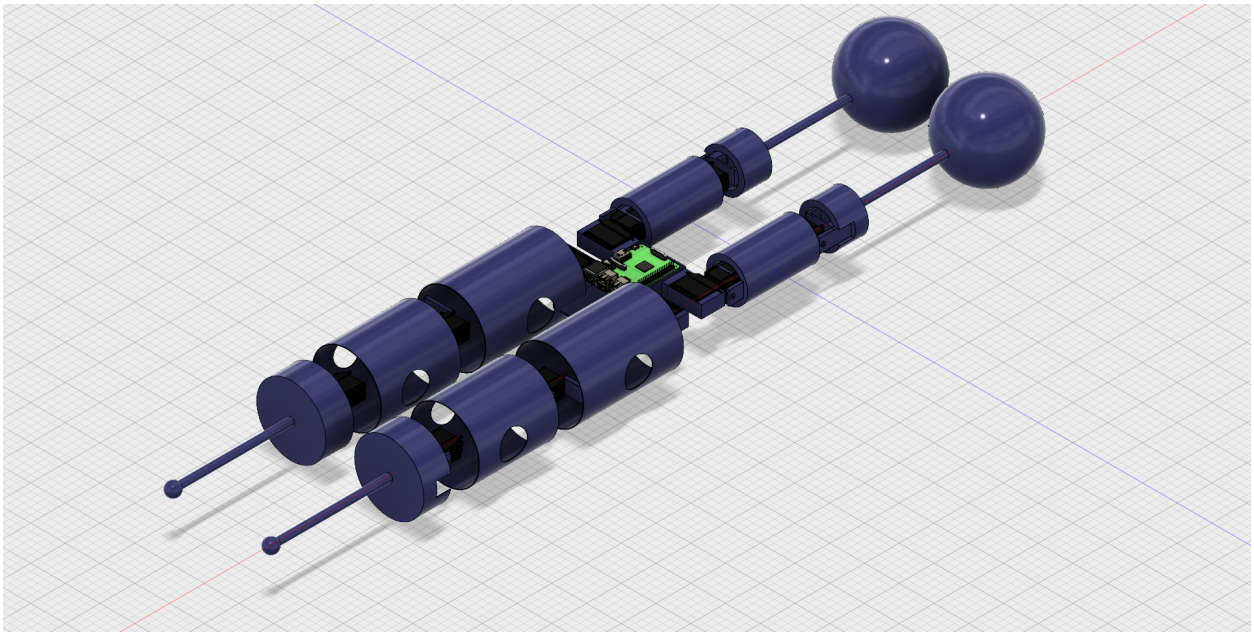
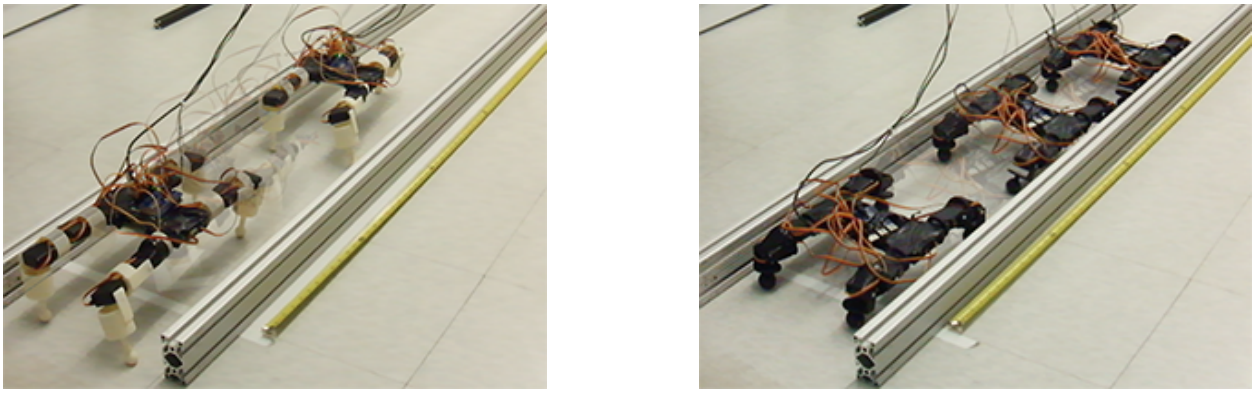


Figure 41: ***The `automated` modified robot design produced using gradient descent and no human supervision. This design has links that are too short to fit the designated actuators and link radii that are too wide to achieve a reasonable range of motion.***

Parameter	`initial`	`automated`	`supervised`
$l_{\text{Front,Hip}}$	62	137	54
$l_{\text{Front,ULeg}}$	72	96	53
$l_{\text{Front,LLeg}}$	140	189	55
$l_{\text{Hind,Hip}}$	62	1	54
$l_{\text{Hind,ULeg}}$	72	81	53
$l_{\text{Hind,LLeg}}$	140	106	55
l_{Base}	87	165	72
w_{Base}	64	72	64
h_{Base}	20	34	25
$r_{\text{Front,\{Hip,ULeg,LLeg\}}}$	23	92	23
$r_{\text{Hind,\{Hip,ULeg,LLeg\}}}$	23	37	23
$r_{\text{Front,Foot}}$	20	8	27
$r_{\text{Hind,Foot}}$	20	187	39
$m_{\text{Front,Hip}}$	17	148	12
$m_{\text{Front,ULeg}}$	18	3	12
$m_{\text{Front,\{LLeg,Foot\}}}$	29	1	15
$m_{\text{Front,Hip}}$	17	200	12
$m_{\text{Front,ULeg}}$	18	166	12
$m_{\text{Hind,\{LLeg,Foot\}}}$	29	69	15

Table 7: *Model parametrization for each robot. Masses (m) are in grams and length (l), width (w), radius (r), and height (h) are in mm.*

completed the 20 gait cycles in only three of the five trials. The robot had trouble supporting its own weight during most trials and then fell to its side in the fourth trial; after repeating this failure in the fifth trial, the robot hardware was irreparably damaged—the servo horn-link interface was stripped. The average velocity during the successful trials was 5.23 cm/s, or about one-third of the commanded velocity. The robot shuffled its feet throughout the trials, indicating that it was unable to produce the necessary torque to properly move its long limbs. This discrepancy was readily detected by the morphological modification process, and the violation was then corrected via modifications to the robot parameters (yielding the `supervised` and `unsupervised` models).



(a) ***initial*** **(b)** ***modified***

Figure 42: ***A time-lapse of the*** `initial` ***and*** `modified` ***robots trotting for 20 gait cycles.***

The `supervised` robot trotted at an average distance of 99 cm over 20 gait cycles, or about 12.22 seconds of trotting. The average velocity during the trials was 8.04 cm/s, about one-half the commanded velocity (and 54% faster than the `initial` robot). The discrepancy between actual and desired trotting speeds were attributed to the robot's feet tending to slide at the moment of push-off during locomotion. The robot remained stable and successfully completed all trials.

6.3 Reconfigurable Quadruped

While Section 6.2 focused on making a newly created low-cost, robot functional, this section presents a modified version of the LINKS robot, built with reconfigurable links (see Figure 43) and attempts to discover the best shape for the adjustable robot. It was hypothesized that the peak performance of the robot was limited by its short legs; the robot was rebuilt to adjust between a size smaller than its original structure (see Figure 21a), to limb and body dimensions that were double that of the original robot; original link dimensions are preserved for robot relative size comparison in Figure 46.

Two model update processes were run, starting from two different `initial` morphological parameterizations of the robot in Figure 43: (1) a `small` robot, with the minimum-size morphological parameter settings; and (2) a `large` robot, with the maximum-size morphological parameter settings (see Table 9). The `modified` robot designs were modified from the parameterizations

for the `initial` robot models.

All four robot configurations were tested by comparing their average forward velocity while performing a trotting gait across a two meter length of floor. This metric was used to determine whether *in situ* performance improved at the target gait velocity of 30 cm/s for the `modified` robot over the `initial` robot. It was expected that the `modified` robots would perform much better at the trotting task, as they were modified to respect their actuator limitations for the prescribed gait.

6.3.1 Platform The test platform was a quadrupedal robot built from Dynamixel actuators joined by aluminum bars. The base of the robot was built from T-slotted aluminum framing bars.

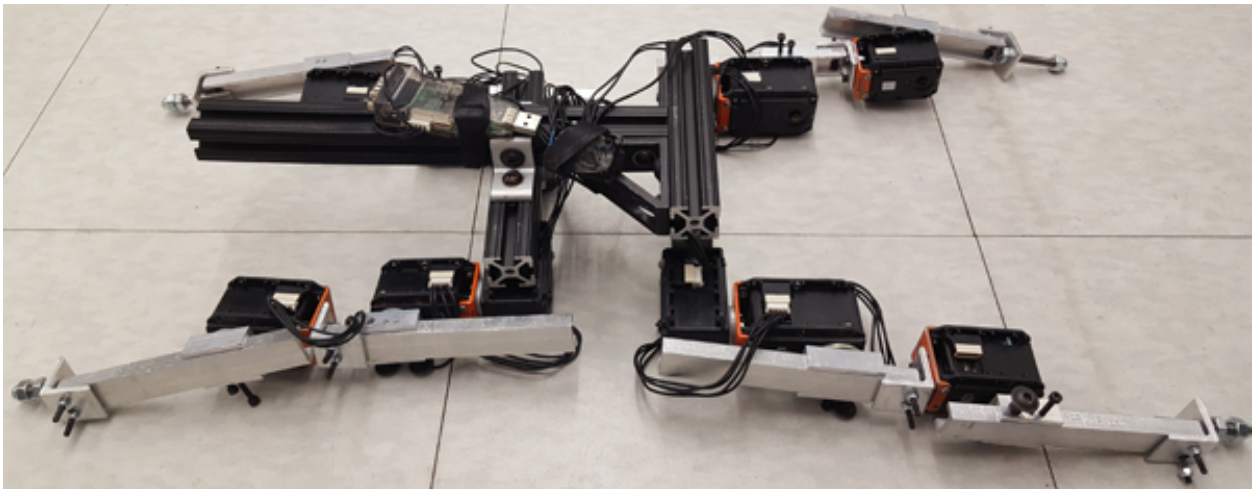


Figure 43: ***The reconfigurable robot in its `initial small` configuration.***

The robot has adjustable limb and body dimensions to permit setting the morphological parameters; relevant morphological parameters and actuator limitations for this particular case are shown in Figure 44.

Update Morphology: *Reconfigurable quadruped*

With morphological parameters $\boldsymbol{p}$:

Parameter (unit)	Links Affected
front, upper limb length, $l_{\text{Front,ULeg}}$ (cm)	{LF, RF}_{ULEG}
front, lower limb length, $l_{\text{Front,LLeg}}$ (cm)	{LF, RF}_{LLEG}
hind, upper limb length, $l_{\text{Hind,ULeg}}$ (cm)	{LH, RH}_{ULEG}
hind, lower limb length, $l_{\text{Hind,LLeg}}$ (cm)	{LH, RH}_{ULEG}
Base length, l_{Base} (cm)	BASE
Base width, w_{Base}(cm)	BASE

Such that *limitations* are not exceeded

Parameter	number of actuators	value
RX-24F Actuator Torque Limits	10	2.6 N·m
RX-24F Actuator Velocity Limits	10	13.195 rad/s
MX-64 Actuator Torque Limits	2	6.0 N·m
MX-64 Actuator Velocity Limits	2	6.597 rad/s

Figure 44: ***Parameters and limits for the reconfigurable quadruped scenario. Limits to robotic hardware considered in the experiment are the stall torque and max velocity of the* RX-24F *and* MX-64 *Dynamixel actuators. The linear torque-speed curves described by these values are drawn in Figure 34.***

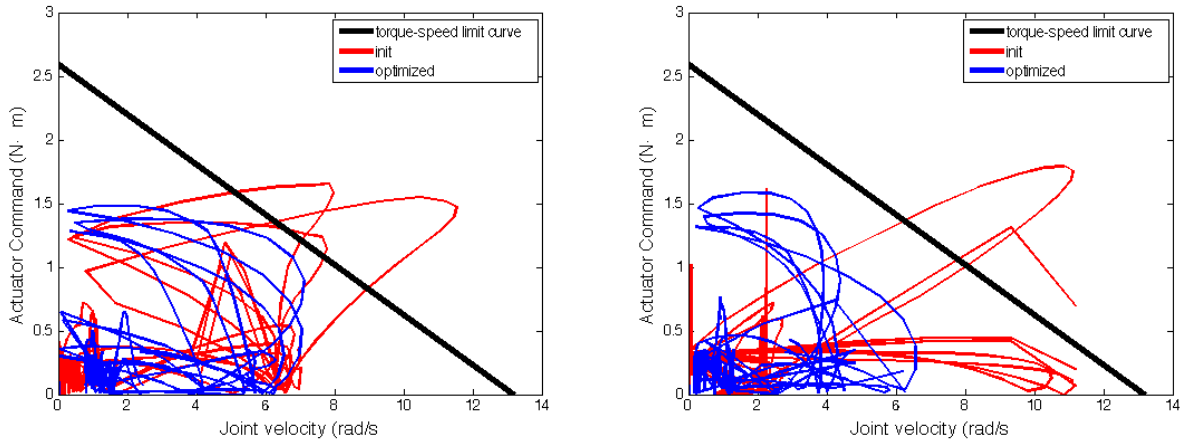
6.3.2 Control policy: gait parameterization A locomotion control policy with gait parameters shown in Table 8 was used both to control the quadruped and to generate task trajectories for the model modification process. These values were chosen as a generally functional set of gait parameters for a 16 cm tall quadruped.

***6.3.3 Results* in situ** This section presents *in situ* results from the morphological modification process for the reconfigurable robot. The `modified` design process began using the `initial` virtual model attempting to perform a trotting gait (see Table 8) at a starting forward velocity of 10 cm/s. The models required adjustments to their kinematic model parameters in order to proceed to the next target velocity (velocities were increased in 10 cm/s increments).

The parameters of each robot's morphology are reported in Table 9 and visualized in Figure 46. The robots *in situ* typically managed to locomote at about half the speed of their virtual counterparts with the same control policy. Like the previous 3D-printed robot, much of this discrepancy in speed

Quadrupedal Gait Parameters	
Parameter (unit)	Value
forward velocity (cm/s)	30
base linear offset (cm)	$\{0, 0, 15\}$
base orientation offset (rad)	$\{0, 0, 0\}$
step height (cm)	2
stance length (cm)	30
stance width (cm)	20
gait duration (sec)	0.3
liftoff timing (% gait duration)	$\{25, 75, 25, 75\}$
duty factor (% gait duration)	$\{60, 60, 60, 60\}$

Table 8: *Gait parameters for the walking task performed by the reconfigurable robot.*



(a) ***small*** **robot** **(b)** ***large*** **robot**

Figure 45: ***The evolution of the trajectory with respect to the torque-speed limit witness function boundary between the*** `initial` ***and*** `modified` ***robot designs when following a trotting gait at 30 cm/s. The*** `initial` ***designs cross the witness function boundary, while the*** `modified` ***designs do not.***

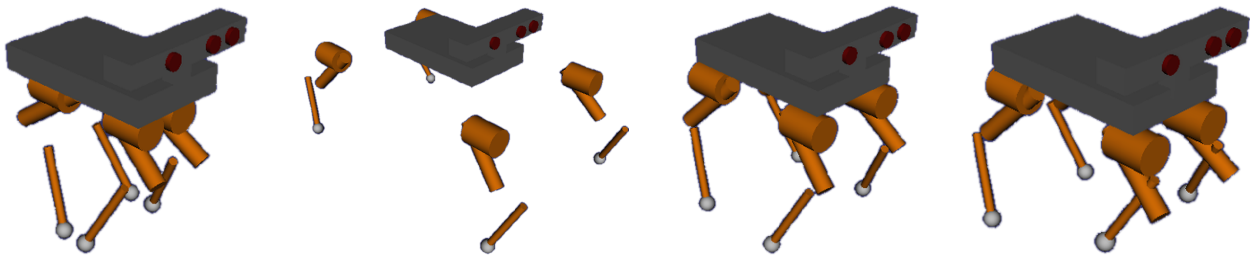
can be attributed to the robot's feet tending to slide at the moment of push-off during locomotion *in situ*. The `initial large` robot was unable to walk forward at all *in situ*. Excessive strain on its joints caused that robot to sag and scuff its knees against the floor as it attempted to walk forward; this led the robot to veer to the side of the testing track.

	Robot			
Parameter	`initial small`	`initial large`	`modified small`	`modified large`
$l_{\text{Front,ULeg}}$ (cm)	5.715	13.335	7.3	11.8
$l_{\text{Front,LLeg}}$ (cm)	9.715	13.335	7.5	5.7
$l_{\text{Hind,ULeg}}$ (cm)	5.715	13.335	8.6	8.0
$l_{\text{Hind,LLeg}}$ (cm)	9.715	13.335	8.9	8.5
l_{Base} (cm)	7.112	30.734	10.9	10.3
w_{Base} (cm)	5.080	23.495	10.6	9.895
Performance				
velocity, *in sim* (cm/s)	< 10	< 10	40	40
velocity, *in situ* (cm/s)	14.7	–	17.7	9.8

Table 9: *Model parametrization and performance of each configuration of the reconfigurable robot. Definitions of these variables are provided in Figure 44*

The modification process updated the parameters to decrease the stress on the actuators by reducing sustained and maximum outputs; the plot in Figure 45 depicts how the morphological adjustment process moved the parameters such that the torque-speed constraint was respected. The average trotting speed of each robot before and after modification is reported in Table 9, and images from physical tests are provided in in Figure 47.

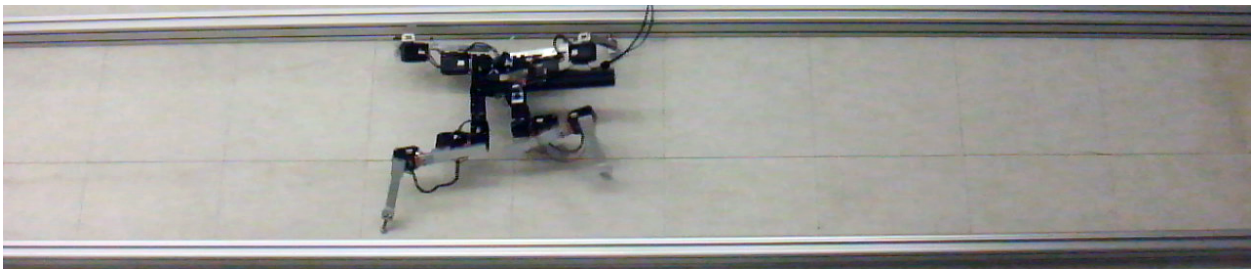
The user controls the scaling of the direction of steepest descent by scaling factor α (see §5.2.2); they can choose which parameters receive greater emphasis in each update to speed the modification process. Small adjustments to α helped steer model updates away from less desirable morphologies. Having a human in the loop was a simple method of detecting when the configuration was approaching infeasible or undesirable parameters. Otherwise, the process required very little input from the user. The `initial small` and `large` robots were updated 24 and 17 times, respectively, before reaching their `modified` states where no further progress could be made (because the robot parameters had converged to a local performance optimum where no



(a) `initial small` (b) `initial large` (c) `modified small` (d) `modified large`

Figure 46: ***The `initial` and `modified` robot designs resulting from a human-in-the-loop design process with gradient descent directions suggested to the designer. To provide a sense of scale for the updated morphological designs, model kinematics were updated while visualization of the original robot links remained fixed. Visualizing these disparately sized robots with the same geometries leads to large gaps or overlaps between links, as seen here.***

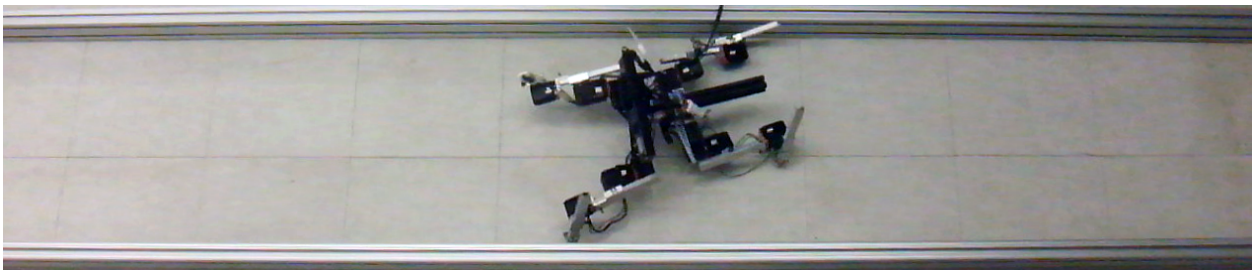
nearby configurations could achieve a faster trotting speed).



(a) ***initial small***

(b) ***initial large***

(c) ***modified small***



(d) ***modified large***

Figure 47: ***Pictures from* in situ *testing with each robot design. The images were captured after 10 seconds of locomotion. The robots are moving from the left side of each image to the right; robots that are further right walked faster than robots that are further left. Each tile is approximately 0.3 meters on a side.***

6.4 Conclusion

This chapter presented demonstrations of a morphological modification process that significantly improved the real-world performance of the tested robots. A substantial increase in maximum locomotion speed was observed between initial and post-modification morphologies during virtual

and *in situ* trails for both robot prototyping scenarios. Design modifications made to the robots in this chapter permitted low-cost robots that initially performed poorly or entirely broke-apart to locomote successfully. The modified robots exhibited substantially more durable behavior than the robots built with the initial designs.

Catastrophic failures such as limb dislocation and veering off the testing track were not observed on the modified robots as they were with the initial robot designs; this supports the hypothesis that the presented model modification approach can be used to combat various divergent behaviors detected by the particle traces and control policy falsification procedures from Chapters 3 and 4. This presented robot design tool could facilitate maximizing robot performance; it discovers what factors limit a robot's ability to perform a task or cause a robot to fail unexpectedly and then mitigate the effects that those limitations have on the behavior of the robot *in situ*.

7 Inverse Dynamics with Contact

This dissertation focuses specifically on robots that physically interact with their environment via contact (i.e., manipulation and locomotion). When contact is expected but does not occur or when contact is not expected but does occur, robot behavior diverges from plan, often disastrously. Chapters 3 and 4 presented and tested a method of detecting such behavioral divergence through virtual stress-testing. Chapters 5 and 6 then presented and tested a method of mitigating the occurrence of these behavioral divergences through virtual robot model modification. In order to provide predictions of divergent behavior and design modifications that are applicable to robots *in situ*, the presented techniques are predicated on the accurate simulation of robotic systems and precise control strategies such as *inverse dynamics* which is used extensively in robotics and biomechanics applications.

In manipulator and legged robots, inverse dynamics can form the basis of an effective nonlinear control strategy by providing a robot with both accurate positional tracking and active compliance. In biomechanics applications, inverse dynamics control can approximately determine the net torques applied at anatomical joints that correspond to an observed motion. In the context of robot control, inverse dynamics requires knowledge of all contact forces acting on the robot. Contact is a governing factor for the movement of legged robots about their environments and for the manner in which robot hands pick up, move, operate, and otherwise manipulate objects in their environment. Measuring such forces is limited by the ability to instrument surfaces and filter force readings, and such filtering effectively delays force sensing to a degree unacceptable for real-time operation. An alternative is to use contact force predictions, for which reasonable agreement between models and reality have been observed (see, e.g., Aukes & Cutkosky 2013). Formulating such approaches is technically challenging, however, because the actuator forces are generally coupled to the contact forces, requiring simultaneous solution. Inverse dynamics approaches that simultaneously compute contact forces exist in literature. Although these approaches were developed without incorporating all of the established modeling aspects (like complementarity) and address-

ing all of the technical challenges (like inconsistent configurations) of hard contact, these methods have been demonstrated performing effectively on real robots. In contrast, *this chapter focuses on formulating inverse dynamics with these established modeling aspects—which allows forward and inverse dynamics models to match—and addresses the technical challenges, including solvability*.

Section 7.1 describes background in rigid body dynamics and rigid contact, as well as related work in inverse dynamics with contact and friction. The implementation of three disparate inverse dynamics formulations are then presented in Sections 7.3, 7.4, and 7.5. Each implementation was aimed to: (1) successfully control a robot through its assigned task; (2) mitigate torque chatter from indeterminacy; (3) evenly distribute contact forces between active contacts; (4) speed computation so that the implementation can be run at realtime on standard hardware. Section 7.3 presents an inverse dynamics formulation with contact force prediction that utilizes the non-impacting rigid contact model (to be described in Section 7.1.4) with no-slip frictional constraints. Section 7.4 presents an inverse dynamics formulation with contact force prediction that utilizes the non-impacting rigid contact model with Coulomb friction constraints. The problem of mitigating torque chatter from indeterminate contact configurations is shown to be no harder than NP-hard. Section 7.5 presents an inverse dynamics formulation that uses a rigid impact model and permits the contact force prediction problem to be convex. This convexity will allow us to mitigate torque chatter from indeterminacy.

Section 7.6 describes experimental setups for assessing the inverse dynamics formulations in the context of simulated robot control along multiple dimensions: accuracy of trajectory tracking; contact force prediction accuracy; general locomotion stability; and computational speed. Tests are performed on terrains with varied friction and compliance. Presented controllers are compared against both PID control and inverse dynamics control with sensed contact and perfectly accurate virtual sensors. Assessment under both rigid and compliant contact models permits both exact and in-the-limit verification that controllers implementing these inverse dynamics approaches for control work as expected. These experiments also examine behavior when modeling assumptions break down. Section 7.7 analyzes the findings from these experiments. Appendix E presents a

chart overviewing the advantages and disadvantages of the presented inverse dynamics algorithms to guide a roboticist toward selecting the best algorithm for their application

7.0.1 Invertibility of the rigid contact model An obstacle to such a formulation has been the claim that the rigid contact model is not invertible (Todorov, 2014), implying that inverse dynamics is unsolveable for multi-rigid bodies subject to rigid contact. If forces on the multi-body *other than contact forces* at state $\{ \boldsymbol{q}, \dot{\boldsymbol{q}} \}$ are designated $\boldsymbol{x}$ and contact forces are designated $\boldsymbol{y}$, then the rigid contact model (to be described in detail in Section 7.1.4) yields the relationship $\boldsymbol{y} = f_{\boldsymbol{q},\dot{\boldsymbol{q}}}(\boldsymbol{x})$. It is then true that there exists no left inverse $g(.)$ of f that provides the mapping $\boldsymbol{x} = g_{\boldsymbol{q},\dot{\boldsymbol{q}}}(\boldsymbol{y})$ for $\boldsymbol{y} = f_{\boldsymbol{q},\dot{\boldsymbol{q}}}(\boldsymbol{x})$. However, this chapter will show that there does exist a right inverse $h(.)$ of f such that, for $\boldsymbol{h}_{\boldsymbol{q},\dot{\boldsymbol{q}}}(\boldsymbol{y}) = \boldsymbol{x}$, $f_{\boldsymbol{q},\dot{\boldsymbol{q}}}(\boldsymbol{x}) = \boldsymbol{y}$, and Section 7.4 shows that this mapping is computable in expected polynomial time. This chapter will use this mapping to formulate inverse dynamics approaches for rigid contact with both no-slip constraints and frictional surface properties.

7.0.2 Indeterminacy in the rigid contact model The rigid contact model is also known to be susceptible to the problem of contact indeterminacy, the presence of multiple equally valid solutions to the contact force-acceleration mapping. This indeterminacy is the factor that prevents strict invertibility and which, when used for contact force predictions in the context of inverse dynamics, can result in torque chatter that is potentially destructive for physically situated robots. Section 7.4.3 shows that computing a mapping from accelerations to contact forces that evolves without harmful torque chatter is no worse than NP-hard in the number of contacts modeled for Coulomb friction and can be calculated in polynomial time for the case of infinite (no-slip) friction.

This chapter also describes a computationally tractable approach for mitigating torque chatter that is based upon a rigid contact model without complementarity conditions (see Sections 7.1.4 and 7.1.4). The model appears to produce reasonable predictions: Anitescu (2006); Drumwright & Shell (2010); Todorov (2014) have all used the model within simulation and physical artifacts have yet to become apparent.

This chapter will assess these inverse dynamics algorithms in the context of controlling a virtual

locomoting robot and a fixed-base manipulator robot. For points of comparison the performance of error feedback and inverse dynamics controllers with virtual contact force sensors are examined with respect to the implemented controllers. A rating of the "performance" of each controller will consider smoothness of torque commands, trajectory tracking accuracy, locomotion performance, and computation time.

7.0.3 Contributions This chapter provides the following contributions:

- Proof that the coupled problem of computing inverse dynamics-derived torques and contact forces under the rigid body dynamics, non-impacting rigid contact, and Coulomb friction models (with linearized friction cone) is solvable in expected polynomial time.
- An algorithm that computes inverse dynamics-derived torques without torque chatter under the rigid body dynamics model and the rigid contact model *assuming no slip along the surfaces of contact*, in expected polynomial time.
- An algorithm that yields inverse dynamics-derived torques without torque chatter under the rigid body dynamics model and the rigid, non-impacting contact model with Coulomb friction in exponential time in the number of points of contact, and hence a proof that this problem is no harder than NP-hard.
- An algorithm that computes inverse dynamics-derived torques without torque chatter under the rigid body dynamics model and a rigid contact model with Coulomb friction but does not enforce complementarity conditions (again, see Sections 7.1.4 and 7.1.4), in expected polynomial time.

These algorithms differ in their operating assumptions. For example, the algorithms that enforce normal complementarity (to be described in Section 7.1.4) assume that all contacts are non-impacting; similarly, the algorithms that do not enforce complementarity assume that bodies are impacting at *one or more* points of contact. As will be explained in Section 7.2, control loop period endpoint times do not necessarily coincide with contact event times, so a single algorithm

must deal with both impacting and non-impacting contacts. It is an open problem of the effects of enforcing complementarity when it should not be enforced, or vice versa. The algorithms also possess various computational strengths. As results of these open problems and varying computational strengths, I will present multiple algorithms to the reader as well as a guide (see Appendix E) that details task use cases for these controllers.

7.1 Background and related work

This section surveys the independent parts that are combined to formulate algorithms for calculating inverse dynamics torques with simultaneous contact force computation. Section 7.1.1 discusses complementarity problems, a domain outside the purview of typical roboticists. Section 2.5 introduces the rigid body dynamics model for Newtonian mechanics under generalized coordinates. Section 7.1.4 covers the rigid contact model, and unilaterally constrained contact. Sections 7.1.4 –7.1.4 show how to formulate constraints on the rigid contact model to account for Coulomb friction and no-slip constraints. Section 7.1.4 describe an algebraic impact model that will form the basis of one of the inverse dynamics methods. Section 7.1.5 describes the phenomenon of "indeterminacy" in the rigid contact model. Lastly, Sections 7.1.6 and 7.1.7 discusses other work relevant to inverse dynamics with simultaneous contact force computation.

7.1.1 Complementarity problems Complementarity problems are a particular class of mathematical programming problems often used to model hard and rigid contacts. A nonlinear complementarity problem (NCP) is composed of three nonlinear constraints (Cottle et al., 1992), which taken together constitute a complementarity condition:

$$\boldsymbol{x} \geq \mathbf{0} \tag{26}$$

$$f(\boldsymbol{x}) \geq \mathbf{0} \tag{27}$$

$$\boldsymbol{x}^{\mathsf{T}} f(\boldsymbol{x}) = 0 \tag{28}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$. Henceforth, the following shorthand will be used to denote a complementarity constraint:

$$0 \leq a \perp b \geq 0 \tag{29}$$

which signifies that $a \geq 0, b \geq 0$, and $a \cdot b = 0$.

A LCP, or linear complementarity problem $(\boldsymbol{r}, \mathbf{Q})$, where $\boldsymbol{r} \in \mathbb{R}^n$ and $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$, is the linear version of this problem:

$$\begin{aligned} \boldsymbol{w} &= \mathbf{Q}\boldsymbol{z} + \boldsymbol{r} \\ \boldsymbol{w} &\geq \mathbf{0} \\ \boldsymbol{z} &\geq \mathbf{0} \\ \boldsymbol{z}^{\mathsf{T}}\boldsymbol{w} &= 0 \end{aligned}$$

for unknowns $\boldsymbol{z} \in \mathbb{R}^n, \boldsymbol{w} \in \mathbb{R}^n$.

Theory of LCPs has been established to a greater extent than for NCPs. For example, theory has indicated certain classes of LCPs that are solvable, which includes both determining when a solution does not exist and computing a solution, based on properties of the matrix $\mathbf{Q}$ (above). Such classes include positive definite matrices, positive semi-definite matrices, P-matrices, and Z-matrices, to name only a few; (Murty, 1988; Cottle et al., 1992) contain far more information on complementarity problems, including algorithms for solving them. Given that the knowledge of NCPs (including algorithms for solving them) is still relatively thin, this chapter will relax NCP instances that correspond to contacting bodies to LCPs using a common practice, linearizing the friction cone.

Duality theory in optimization establishes a correspondence between LCPs and quadratic programs (see Cottle et al., 1992, Pages 4 and 5) via the Karush-Kuhn-Tucker conditions; for example, positive semi-definite LCPs are equivalent to convex QPs. Algorithms for quadratic programs (QPs) can be used to solve LCPs, and vice versa.

7.1.2 Relationship between LCPs and MLCPs

This section describes the *mixed linear complimentarity problem* (MLCP) and its relationship to the "pure" LCP.

Assume the LCP $(\boldsymbol{r}, \mathbf{Q})$ for $\boldsymbol{r} \in \mathbb{R}^a$ and $\mathbf{Q} \in \mathbb{R}^{a \times a}$:

$$\boldsymbol{w} = \mathbf{Q}\boldsymbol{z} + \boldsymbol{r} \quad \boldsymbol{w} \geq \mathbf{0} \quad \boldsymbol{z} \geq \mathbf{0} \quad \boldsymbol{z}^\mathsf{T}\boldsymbol{w} = 0 \tag{30}$$

for unknown vectors $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{R}^a$. A mixed linear complementarity problem (MLCP) is defined by the matrices $\mathbf{A} \in \mathbb{R}^{p \times s}$, $\mathbf{C} \in \mathbb{R}^{p \times t}$, $\mathbf{D} \in \mathbb{R}^{r \times s}$, $\mathbf{B} \in \mathbb{R}^{r \times t}$, $\boldsymbol{x} \in \mathbb{R}^s$, $\boldsymbol{y} \in \mathbb{R}^t$, $\boldsymbol{g} \in \mathbb{R}^p$, and $\boldsymbol{h} \in \mathbb{R}^r$ (where $p = s$ and $r = t$) and is subject to the following constraints:

$$\mathbf{A}\boldsymbol{x} + \mathbf{C}\boldsymbol{z} + \boldsymbol{g} = \mathbf{0} \tag{31}$$

$$\mathbf{D}\boldsymbol{x} + \mathbf{B}\boldsymbol{z} + \boldsymbol{h} \geq \mathbf{0} \tag{32}$$

$$\boldsymbol{z} \geq \mathbf{0} \tag{33}$$

$$\boldsymbol{z}^\mathsf{T}(\mathbf{D}\boldsymbol{x} + \mathbf{B}\boldsymbol{z} + \boldsymbol{h}) = 0 \tag{34}$$

The $\boldsymbol{x}$ variables are unconstrained, while the $\boldsymbol{z}$ variables must be non-negative. If $\mathbf{A}$ is non-singular, the unconstrained variables can be computed as:

$$\boldsymbol{x} = -\mathbf{A}^{-1}(\mathbf{C}\boldsymbol{z} + \boldsymbol{g}) \tag{35}$$

Substituting $\boldsymbol{x}$ into Equations 31–153 yields the pure LCP $(\boldsymbol{r}, \mathbf{Q})$:

$$\mathbf{Q} \equiv \mathbf{B} - \mathbf{D}\mathbf{A}^{-1}\mathbf{C} \tag{36}$$

$$\boldsymbol{r} \equiv \boldsymbol{h} - \mathbf{D}\mathbf{A}^{-1}\boldsymbol{g} \tag{37}$$

A solution $(\boldsymbol{z}, \boldsymbol{w})$ to this LCP obeys the relationship $\mathbf{Q}\boldsymbol{z} + \boldsymbol{r} = \boldsymbol{w}$; given $\boldsymbol{z}$, $\boldsymbol{x}$ is determined via Equation 35, solving the MCLP.

7.1.3 The multi-body This chapter centers around *a multi-body*, which is the system of rigid bodies (see §2.5) to which inverse dynamics is applied. The multi-body may come into contact with "fixed" parts of the environment (*e.g.*, solid ground) which are sufficiently modeled as non-moving bodies— this is often the case when simulating locomotion. Alternatively, the multi-body may contact other bodies, in which case effective inverse dynamics will require knowledge of those bodies' kinematic and dynamic properties — necessary for manipulation tasks.

The articulated body approach can be extended to a multi-body to account for physically interacting with movable rigid bodies by appending the six degree-of-freedom velocities ($\boldsymbol{v}_{\text{cb}}$) and external wrenches ($\boldsymbol{f}_{\text{cb}}$) of each contacted rigid body to the velocity and external force vectors and by augmenting the generalized inertia matrix (M) similarly:

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_{\text{robot}}{}^{\mathsf{T}} & {}^{+}\boldsymbol{v}_{\text{cb}}{}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \tag{38}$$

$$\boldsymbol{f}_{\text{ext}} = \begin{bmatrix} \boldsymbol{f}_{\text{robot}}{}^{\mathsf{T}} & \boldsymbol{f}_{\text{cb}}{}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \tag{39}$$

$$\mathrm{M} = \begin{bmatrix} \mathrm{M}_{\text{robot}} & 0 \\ 0 & \mathrm{M}_{\text{cb}} \end{bmatrix} \tag{40}$$

Without loss of generality, this chapter will hereafter consider only a single multi-body in contact with a static environment (excepting an example with a manipulator arm grasping a box in Section 7.6).

7.1.4 Rigid contact model This section will summarize existing theory of modeling non-impacting rigid contact and draws from Stewart & Trinkle (1996); Trinkle et al. (1997); Anitescu & Potra (1997). Let us define a set of gap functions $\phi_i(\boldsymbol{x})$ (for $i = 1, \ldots, q$), where gap function i gives the signed distance between a link of the robot and another rigid body (part of the environment, another link of the robot, an object to be manipulated, *etc.*)

The notation in this chapter assumes independent coordinates $\boldsymbol{x}$ (and velocities $\boldsymbol{v}$ and accelerations $\dot{\boldsymbol{v}}$), and that generalized forces and inertias are also given in minimal coordinates.

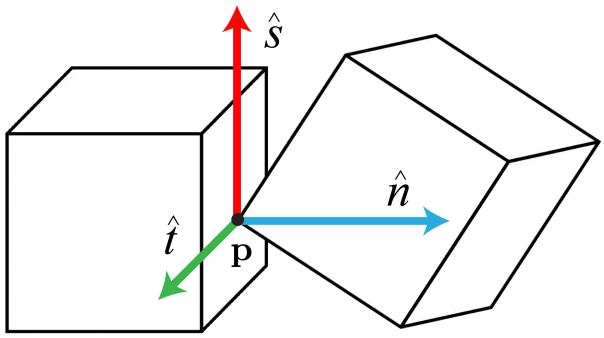


Figure 48: ***The contact frame consisting of $\hat{n}$, $\hat{s}$, and , $\hat{t}$ vectors corresponding to the normal, first tangential, and second tangential directions (for 3D) to the contact surface.***

The gap functions return a positive real value if the bodies are separated, a negative real value if the bodies are geometrically intersecting, and zero if the bodies are in a "kissing" configuration. The rigid contact model specifies that bodies never overlap, *i.e.*:

$$\phi_i(\boldsymbol{x}) \geq 0 \quad \text{for } i = 1, \ldots, q \tag{41}$$

One or more points of contact between bodies is determined for two bodies in a kissing configuration ($\phi_i(\boldsymbol{x}) = 0$). *For clarity of presentation, it was assumes that each gap function corresponds to exactly one point of contact (even for disjoint bodies), so that* $n = q$. In the absence of friction, the constraints on the gap functions are enforced by forces that act along the contact normal. Projecting these forces along the contact normal yields scalars $f_{N_1}, \ldots, f_{N_n}$. The forces should be compressive (*i.e.*, forces that can not pull bodies together), which is denoted by restricting the sign of these scalars to be non-negative:

$$f_{N_i} \geq 0 \quad \text{for } i = 1, \ldots, n \tag{42}$$

A *complementarity* constraint keeps frictional contacts from doing work: when the constraint is inactive ($\phi_i > 0$) no force is applied and when force is applied, the constraint must be active ($\phi_i = 0$). This constraint is expressed mathematically as $\phi_i \cdot f_{N_i} = 0$. All three constraints can be

combined into one equation using the notation in Section 7.1.1:

$$0 \leq f_{N_i} \perp \phi_i(\boldsymbol{x}) \geq 0 \quad \text{for } i = 1, \ldots, n \tag{43}$$

These constraints can be differentiated with respect to time to yield velocity-level or acceleration-level constraints suitable for expressing the differential algebraic equations (DAEs), as an index 1 DAE:

$$0 \leq f_{N_i} \perp \dot{\phi}_i(\boldsymbol{x}) \geq 0 \text{ if } \phi_i = 0 \quad \text{for } i = 1, \ldots, n \tag{44}$$

$$0 \leq f_{N_i} \perp \ddot{\phi}_i(\boldsymbol{x}) \geq 0 \text{ if } \phi_i = \dot{\phi}_i = 0 \text{ for } i = 1, \ldots, n \tag{45}$$

Modeling Coulomb friction Dry friction is often simulated using the Coulomb model, a relatively simple, empirically derived model that yields the approximate outcome of sophisticated physical interactions. Coulomb friction covers two regimes: sticking/rolling friction (where the tangent velocity at a point of contact is zero) and sliding friction (nonzero tangent velocity at a point of contact). Rolling friction is distinguished from sticking friction by whether the bodies are moving relative to one another other than at the point of contact.

There are many phenomena Coulomb friction does not model, including "drilling friction" (it is straightforward to augment computational models of Coulomb friction to incorporate this feature, as seen in Leine & Glocker, 2003), the Stribeck effect (Stribeck, 1902), and viscous friction, among others. This chapter focuses only on Coulomb friction, because it captures important stick/slip transitions and uses only a single parameter; the LuGRe model (Do et al., 2007), for example, is governed by seven parameters, making system identification tedious.

Coulomb friction uses a unitless friction coefficient, commonly denoted μ. If the tangent velocities and accelerations are defined in 3D frames (located at the i^{th} point of contact) as v_{S_i}/v_{T_i} and a_{S_i}/a_{T_i}, respectively, and the tangent forces as f_{S_i} and f_{T_i}, then the sticking/rolling constraints which are applicable exactly when $0 = v_{S_i} = v_{T_i}$, can be expressed via the Fritz-John optimality

conditions (Mangasarian & Fromovitz, 1967; Trinkle et al., 1997):

$$0 \leq \mu_i^2 f_{N_i}^2 - f_{S_i}^2 - f_{T_i}^2 \perp a_{S_i}^2 + a_{T_i}^2 \geq 0 \tag{46}$$

$$\mu_i f_{N_i} a_{S_i} + f_{S_i} \sqrt{a_{S_i}^2 + a_{T_i}^2} = 0 \tag{47}$$

$$\mu_i f_{N_i} a_{T_i} + f_{T_i} \sqrt{a_{S_i}^2 + a_{T_i}^2} = 0 \tag{48}$$

These conditions ensure that the friction force lies within the friction cone (Equation 46) and that the friction forces push against the tangential acceleration (Equations 47–48).

In the case of sliding at the i^{th} contact ($v_{S_i} \neq 0$ or $v_{T_i} \neq 0$), the constraints become:

$$\mu_i^2 f_{N_i} - f_{S_i}^2 - f_{T_i}^2 \geq 0 \tag{49}$$

$$\mu_i f_{N_i} v_{S_i} + f_{S_i} \sqrt{v_{S_i}^2 + v_{T_i}^2} = 0 \tag{50}$$

$$\mu_i f_{N_i} v_{T_i} + f_{T_i} \sqrt{v_{S_i}^2 + v_{T_i}^2} = 0 \tag{51}$$

Note that this case is only applicable if $v_{S_i}^2 + v_{T_i}^2 > 0$, so there is no need to include such a constraint in Equation 49 (as was necessary in Equation 46).

The rigid contact model with Coulomb friction is subject to *inconsistent configurations* (Stewart, 2000a), exemplified by Painlevé's Paradox (Painlevé, 1895), in which impulsive forces may be necessary to satisfy all constraints of the model *even when the bodies are not impacting*. The acceleration-level dynamics can be approximated using finite differences; a first order approximation is often used (see, *e.g.*, Posa & Tedrake, 2012), which moves the problem to the velocity/impulsive force domain. Such an approach generally uses an *algebraic collision law* (see Chatterjee & Ruina, 1998) to model all contacts, both impacting, as inelastic impacts; typical "time stepping methods" (Moreau, 1983) for simulating dynamics often treat the generalized coordinates as constant over the small timespan of contact/impact (*i.e.*, a first order approximation); see, *e.g.*, Stewart & Trinkle (2000). Stewart has shown that this approximation converges to the solution of the continuous time formulation as the step size tends to zero (1998).

Upon moving to the velocity/impulsive force domain, Equations 46–51 require a slight transformation to the equations:

$$0 \leq \mu_i^2 f_{N_i}^2 - f_{S_i}^2 - f_{T_i}^2 \perp v_{S_i}^2 + v_{T_i}^2 \geq 0 \tag{52}$$

$$\mu f_{N_i} v_{S_i} + f_{S_i} \sqrt{v_{S_i}^2 + v_{T_i}^2} = 0 \tag{53}$$

$$\mu f_{N_i} v_{T_i} + f_{T_i} \sqrt{v_{S_i}^2 + v_{T_i}^2} = 0 \tag{54}$$

and there is no longer separate consideration of sticking/rolling and slipping contacts.

No-slip constraints If the Coulomb friction constraints are replaced by no-slip constraints, which is a popular assumption in legged locomotion research, one must also use the discretization approach; without permitting impulsive forces, slip can occur even with infinite friction (Lynch & Mason, 1995). The no-slip constraints are then simply $v_{S_i} = v_{T_i} = 0$ (replacing Equations 52–54).

Model for rigid, non-impacting contact with Coulomb friction The model of rigid contact with Coulomb friction for two bodies in non-impacting rigid contact at $\boldsymbol{p}$ can be summarized by the following equations:

$$0 \leq f_n \perp a_n \geq 0 \tag{55}$$

$$0 \leq \mu^2 f_n^2 - f_s^2 - f_t^2 \perp \sqrt{v_s^2 + v_t^2} \geq 0 \tag{56}$$

$$0 = \mu f_n v_s + f_s \sqrt{v_s^2 + v_t^2} \tag{57}$$

$$0 = \mu f_n v_t + f_t \sqrt{v_s^2 + v_t^2} \tag{58}$$

$$0 = \mu f_n a_s + f_s \sqrt{a_s^2 + a_t^2} \tag{59}$$

$$0 = \mu f_n a_t + f_t \sqrt{a_s^2 + a_t^2} \tag{60}$$

where f_n, f_s, and f_t are the signed magnitudes of the contact force applied along the normal and two tangent directions, respectively; a_n is the relative acceleration of the bodies normal to the contact surface; and v_s and v_t are the relative velocities of the bodies projected along the two tangent directions. The operator $\perp$ indicates that $\boldsymbol{a} \cdot \boldsymbol{b} = 0$, for vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. Detailed

interpretation of these equations can be found in Trinkle et al. (1997); a summary is presented below. Equation 55 ensures that (1) only compressive forces are applied ($f_n \geq 0$); (2) bodies cannot interpenetrate ($a_n \geq 0$); and (3) no work is done for frictionless contacts ($f_n \cdot a_n = 0$). Equation 56 models Coulomb friction: either the velocity in the contact tangent plane is zero—which allows frictional forces to lie within the friction cone—or the contact is slipping and the frictional forces must lie strictly on the edge of the friction cone. Equations 57 and 58—applicable to sliding contacts (*i.e.*, those for which $v_s \neq 0$ or $v_t \neq 0$)—constrain friction forces to act against the direction of slip, while Equations 59 and 60 constrain frictional forces for rolling or sticking contacts (*i.e.*, those for which $v_s = v_t = 0$) to act against the direction of tangential acceleration.

These equations form a nonlinear complementarity problem (Cottle et al., 1992), and this problem may not possess a solution with nonimpulsive forces due to the existence of inconsistent configurations like Painlevé's Paradox (Stewart, 2000b). This issue led to the movement to the impulsive force/velocity domain for modeling rigid contact, which can provably model the dynamics of such inconsistent configurations.

A separate issue with the rigid contact model is that of indeterminacy, where multiple sets of contact forces and possibly multiple sets of accelerations—or velocities, if an impulse-velocity model is used—can satisfy the contact model equations. The inverse dynamics approaches presented in this chapter, which use rigid contact models, address inconsistency and, given some additional computation, can address indeterminacy (useful for controlled systems).

Contacts without complementarity Complementarity along the surface normal arises from Equation 43 for contacting rigid bodies that are not impacting. For impacting bodies, complementarity conditions are unrealistic (Chatterjee, 1999). Though the distinction between impacting and non-impacting may be clear in free body diagrams and symbolic mathematics, the distinction between the two modes is arbitrary in floating point arithmetic. This arbitrary distinction has led researchers in dynamic simulation, for example, to use one model—either with complementarity or without—for both impacting and non-impacting contact.

Anitescu (2006) described a contact model without complementarity (Equation 43) used for multi-rigid body simulation. Drumwright & Shell (2010) and Todorov (2014) rediscovered this model (albeit with slight modifications, addition of viscous friction, and guarantees of solution existence and non-negativity energy dissipation in the former); Drumwright & Shell (2010) also motivated acceptability of removing the complementarity condition based on the work by Chatterjee (1999). When treated as a simultaneous impact model, the model is consistent with first principles. Additionally, using arguments in Smith et al. (2012), it can be shown that solutions of this model exhibit symmetry. This impact model, under the assumption of inelastic impact—it is possible to model partially or fully elastic impact as well, but one must then consider numerous models of restitution, see, *e.g.*, Chatterjee & Ruina (1998)—will form the basis of the inverse dynamics approach described in Section 7.5.

The model is formulated as the convex quadratic program below. For consistency of presentation with the non-impacting rigid model described in the previous section, only a single impact point is considered.

Complementarity-free impact model (single point of contact)

$$\underset{{}^{+}\boldsymbol{v},\boldsymbol{f}_n,\boldsymbol{f}_s,\boldsymbol{f}_t}{\text{minimize}} \quad \text{dissipate kinetic energy maximally:} \quad \frac{1}{2}{}^{+}\boldsymbol{v}^{\mathsf{T}}\mathbf{M}{}^{+}\boldsymbol{v} \tag{61}$$

$$\text{subject to:} \quad \text{non-interpenetration:} \quad \mathbf{n}{}^{+}\boldsymbol{v} \geq \mathbf{0} \tag{62}$$

$$\text{compressive normal forces} \quad \boldsymbol{f}_n \geq \mathbf{0} \tag{63}$$

$$\text{Coulomb friction:} \quad \mu^2 f_n \geq f_s + f_t \tag{64}$$

$$\text{first-order dynamics:} \quad {}^{+}\boldsymbol{v} = {}^{-}\boldsymbol{v} + \mathbf{M}^{-1}(\mathbf{n}^{\mathsf{T}}\boldsymbol{f}_n + \mathbf{s}^{\mathsf{T}}\boldsymbol{f}_s + \mathbf{t}^{\mathsf{T}}\boldsymbol{f}_t) \tag{65}$$

where f_n, f_s, and f_t are the signed magnitudes of the impulsive forces applied along the normal and two tangent directions, respectively; ${}^{-}\boldsymbol{v} \in \mathbb{R}^m$ and ${}^{+}\boldsymbol{v} \in \mathbb{R}^m$ are the generalized velocities of

the multi-body immediately before and after impact, respectively; $\mathbf{M} \in \mathbb{R}^{m \times m}$ is the generalized inertia matrix of the m degree of freedom multi-body; and $\boldsymbol{n} \in \mathbb{R}^m$, $\boldsymbol{s} \in \mathbb{R}^m$, and $\boldsymbol{t} \in \mathbb{R}^m$ are generalized wrenches applied along the normal and two tangential directions at the point of contact (see Appendix A for further details on these matrices).

The physical interpretation of the impact model is straightforward: it selects impact forces that maximize the rate of kinetic energy dissipation. Finally, it should be noted that rigid impact models do not enjoy the same degree of community consensus as the non-impacting rigid contact models because three types of impact models (algebraic, incremental, and full deformation) currently exist (Chatterjee & Ruina, 1998), because simultaneous impacts and impacts between multi-bodies can be highly sensitive to initial conditions (Ivanov, 1995), and because intuitive physical parameters for capturing all points of the feasible impulse space do not yet exist (Chatterjee & Ruina, 1998), among other issues. These difficulties lead this chapter to consider only inelastic impacts, a case for which the feasible impulse space is constrained.

7.1.5 Contact force indeterminacy In previous work (Zapolsky et al., 2013), we discovered that indeterminacy in the rigid contact model can be a significant problem for controlling quadrupedal robots (and, presumably, hexapods, *etc.*) by yielding torques that switch rapidly between various actuators (torque chatter). The problem can occur in bipedal walkers; for example, Collins et al. (2001) observed instability from rigid contact indeterminacy in passive walkers. Even manipulators may also experience the phenomenon of rigid contact indeterminacy, indicated by torque chatter.

Rigid contact configurations can be indeterminate in terms of forces; for the example of a table with all four legs perfectly touching a ground plane, infinite enumerations of force configurations satisfy the contact model (as discussed in Mirtich, 1996), although the accelerations predicted by the model are unique. Other rigid contact configurations can be indeterminate in terms of predicting different accelerations/velocities through multiple sets of valid force configurations. Two methods of mitigating indeterminacy are presented in this chapter (see Sections 7.3.6 and 7.5.2). Defining a manner by which actuator torques evolve over time, or selecting a preferred distribution

of contact forces may remedy the issues resulting from indeterminacy.

7.1.6 Contact models for inverse dynamics in the context of robot control This section focuses on "hard", as in perfectly rigid, models of contact incorporated into inverse dynamics and whole body control for robotics. Research that has attempted to combine inverse dynamics with compliant contact was not discovered over the course of developing this research (one possible reason for absence of such work is that such compliant models can require significant parameter tuning for accuracy and to prevent prediction of large contact forces).

Mistry et al. (2010) developed a fast inverse dynamics control framework for legged robots in contact with rigid environments under the assumptions that feet do not slip. Righetti et al. (2013) extended this work with a framework that permits quickly optimizing a mixed linear/quadratic function of motor torques and contact forces using fast linear algebra techniques. Hutter & Siegwart (2012) also uses this formulation in an operational space control scheme, simplifying the contact mathematics by assuming contacts are sticking. Mistry et al.; Righetti et al.; Hutter & Siegwart demonstrate effective trajectory tracking performance on quadrupedal robots.

The inverse dynamics approach of Ames (2013) assumes sticking impact upon contact with the ground and immediate switching of support to the new contact, while enforcing a unilateral constraint of the normal forces and predicting no-slip frictional forces.

Kuindersma et al. (2014) use a no-slip constraint but allow for bounded violation of that constraint in order to avoid optimizing over an infeasible or inconsistent trajectory.

Stephens & Atkeson (2010a) incorporate a contact model into an inverse dynamics formulation for dynamic balance force control. Their approach uses a quadratic program (QP) to estimate contact forces quickly on a simplified model of a bipedal robot's dynamics. Newer work by Feng et al. (2013) builds on this by approximating the friction cone with a circumscribed friction pyramid.

Ott et al. (2011) also use an optimization approach for balance, modeling contact to distribute forces among a set of pre-defined contacts to enact a generalized wrench on a simplified model of a biped; their controller seeks to minimize the Euclidian norm of the predicted contact forces

to mitigate slip. In underconstrained cases (where multiple solutions to the inverse dynamics with contact system exist), Saab et al. (2013) and Zapolsky et al. (2013) use a multi-phase QP formulation for bipeds and quadrupeds, respectively. Zapolsky et al. mitigates the indeterminacy in the rigid contact model by selecting a solution that minimizes total actuator torques, while Saab et al. use the rigid contact model in the context of cascades of QPs to perform several tasks in parallel (*i.e.*, whole body control). The latter work primarily considers contacts without slip, but does describe modifications that would incorporate Coulomb friction (inconsistent and indeterminate rigid contact configurations are not addressed). Todorov (2014) uses the same contact model (to be described below) but without using a two-stage solution; that approach uses regularization to make the optimization problem strictly convex (yielding a single, globally optimal solution). None of Saab et al.; Zapolsky et al.; Todorov utilize the *complementarity constraint* (*i.e.*, $f_N \perp \phi$ in Equation 43) in their formulation. Zapolsky *et al.* and Todorov motivate dropping this constraint in favor of maximizing energy dissipation through contact, an assumption that they show performs reasonably in practice (Drumwright & Shell, 2010; Todorov, 2011).

7.1.7 Contact models for inverse dynamics in the context of biomechanics

Inverse dynamics is commonly applied in biomechanics to determine approximate net torques at anatomical joints for observed motion capture and force plate data. Standard Newton-Euler inverse dynamics algorithms (as described in Featherstone, 2008) are applied; least squares is required because the problem is overconstrained. Numerous such approaches are found in biomechanics literature, including (Kuo, 1998; Hatze, 2002; Blajer et al., 2007; Bisseling & Hof, 2006; Yang et al., 2007; Van Den Bogert & Su, 2008; Sawers & Hahn, 2010). These force plate based approaches necessarily limit the environments in which the inverse dynamics computation can be conducted.

7.2 Discretized inverse dynamics

Inverse dynamics is discretized in this work because the resolution to rigid contact models both without slip and with Coulomb friction can require impulsive forces even when there are no impacts (see Section 7.1.4). This choice will imply that the dynamics are accurate to only first order,

but that approximation should limit modeling error considerably for typical control loop rates (Zapolsky & Drumwright, 2015).

As noted above, dynamics are discretized using a first order approximation to acceleration. Thus, the solution to the equation of motion $\dot{\boldsymbol{v}} = \mathbf{M}^{-1}\boldsymbol{f}$ over $[t_0, t_f]$ is approximated by ${}^{+}\boldsymbol{v} = {}^{-}\boldsymbol{v} + \Delta t\, {}^{-}\mathbf{M}^{-1}\, {}^{-}\boldsymbol{f}$, where $\Delta t = (t_f - t_0)$. The superscript "$^{+}$" is used to denote that a term is evaluated at t_f and the superscript "$^{-}$" is applied to denote that a term is computed at t_0. As examples, the generalized inertia matrix ${}^{-}\mathbf{M}$ is computed at t_0 and the generalized post-contact velocity (${}^{+}\boldsymbol{v}$) is computed at t_f. Hereafter the convention will be adopted that application of a superscript once will indicate implicit evaluation of that quantity at that time thereafter (unless another superscript is applied). For example, matrix $\mathbf{M}$ will continue to be treated as evaluated at t_0 in the remainder of this chapter.

The remainder of this section describes how contact constraints should be determined for discretized inverse dynamics.

7.2.1 Incorporating contact into planned motion The inverse dynamics controller attempts to realize a planned motion. That planned motion must account for pose data and geometric models of objects in the robot's environment. If planned motion is inconsistent with contact constraints, e.g., the robot attempts to push through a wall, undesirable behavior will clearly result. Obtaining accurate geometric data (at least for novel objects) and pose data are presently challenging problems; additional work in inverse dynamics control with predictive contact is necessary to address contact compliance and sensing uncertainty.

7.2.2 Incorporating contact constraints that do not coincide with control loop period endpoint times Contact events—making or breaking contact, transitioning from sticking to sliding or vice versa—do not generally coincide with control loop period endpoint times. Introducing a contact constraint "early", i.e., before the robot comes into contact with an object, will result in a poor estimate of the load on the robot (as the anticipated link-object reaction force will be absent). Introducing a contact constraint "late", i.e., after the robot has already contacted the object, implies

that an impact occurred; it is also likely that actuators attached to the contacted link and on up the kinematic chain are heavily loaded, resulting in possible damage to the robot, the environment, or both. Figure 49 depicts both of these scenarios for a walking bipedal robot.

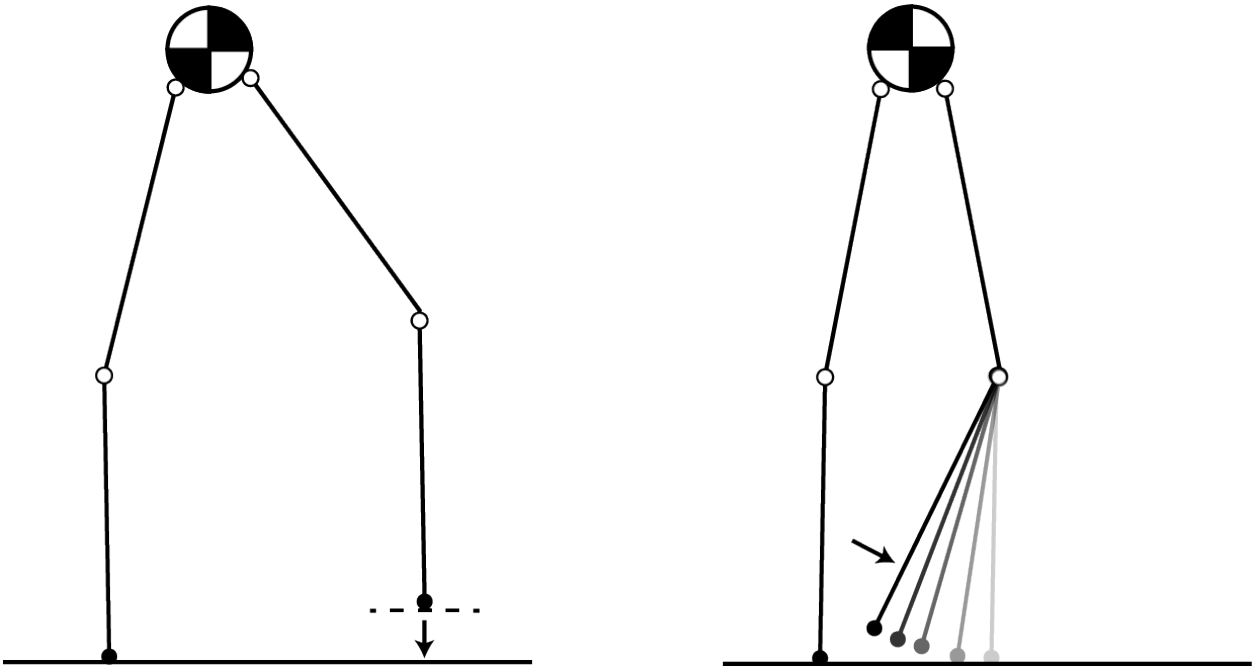


Figure 49: ***If the contact constraint is introduced early (left figure, constraint depicted using dotted line) the anticipated load will be wrong. The biped will pitch forward, possibly falling over in this scenario. If the contact constraint is introduced late, an impact may occur while the actuators are loaded. The biped on the right is moving its right lower leg toward a foot placement; the impact as the foot touches down is prone to damaging the loaded powertrain.***

This problem is addressed by borrowing a *constraint stabilization* (Ascher et al., 1995) approach from Anitescu & Hart (2004), which is itself a form of *Baumgarte Stabilization* (Baumgarte, 1972). Recalling that two bodies are separated by signed distance $\phi(.)$, constraints on velocities are determined such that .

To realize these constraints mathematically, Equation 44 is first converted to a discretized form:

$$0 \leq f_{N_i}(t) \perp \dot{\phi}_i(\boldsymbol{x}(t + \Delta t)) \geq 0 \text{ if } \phi_i(t) = 0 \qquad \text{for } i = 1, \ldots, n \tag{66}$$

This equation specifies that a force is to be found such that applying the force between one of the robot's links and an object, *already in contact at* t, over the interval $[t, t + \Delta t]$ yields a relative

velocity indicating sustained contact or separation at $t + \Delta t$. Next the signed distance between the bodies is incorporated:

$$0 \leq f_{N_i}(t) \perp \dot{\phi}_i(\boldsymbol{x}(t + \Delta t)) \geq -\frac{\phi(\boldsymbol{x}(t))}{\Delta t} \qquad \text{for } i = 1, \ldots, n \tag{67}$$

The removal of the conditional makes the constraint always active. Introducing a constraint of this form means that forces may be applied in some scenarios when they should not be (see Figure 50 for an example). Alternatively, constraints introduced before bodies contact can be contradictory, making the problem infeasible. Existing proofs for time stepping simulation approaches indicate that such issues disappear for sufficiently small integration steps (or, in the context of inverse dynamics, sufficiently high frequency control loops); see Anitescu & Hart (2004), which proves that such errors are uniformly bounded in terms of the size of the time step and the current value of the velocity.

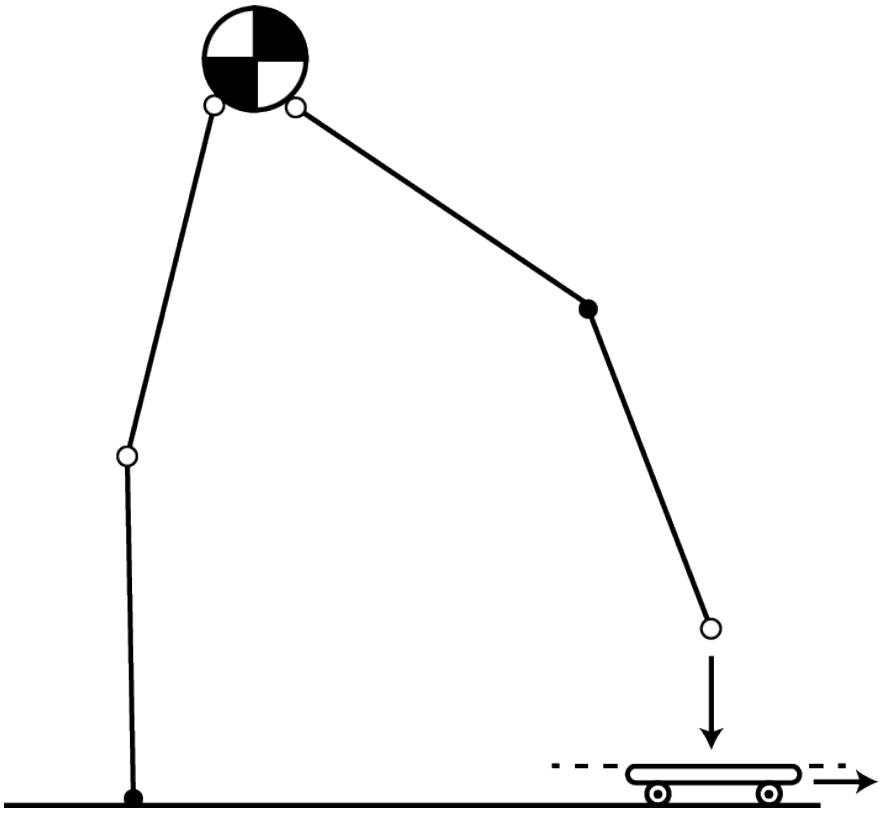


Figure 50: ***An example of a contact constraint determined at time*** t_0 ***(the time of the depicted configuration) that could predict overly constrained motion at*** $t_0 + \Delta t$ ***(the next control loop trigger time) between two disjoint bodies: the right foot and the skateboard. The contact constraint precludes predictions that the foot could move below the dotted line. If the contact force prediction is computed using the current depiction (at*** t_0***) and the skateboard moves quickly to the right such that no contact would occur between the foot and the skateboard at*** $t_0 + \Delta t$***, the correct, contact force (zero) will not be predicted. It should be apparent that these problems disappear as*** $\Delta t \to 0$***, i.e., as the control loop frequency becomes sufficiently high.***

7.2.3 Computing points of contact between geometries Given object poses data and geometric models, points of contact between robot links and environment objects can be computed using closest features. The particular algorithm used for computing closest features is dependent upon both the representation (e.g., implicit surface, polyhedron, constructive solid geometry) and the shape (e.g., sphere, cylinder, box). Algorithms and code can be found in sources like Ericson (2005) and `http://geometrictools.com`. Figure 51 depicts the procedure for determining contact points and normals for two examples: a sphere vs. a half-space and for a sphere vs. a sphere.

For disjoint bodies like those depicted in Figure 51, the contact point can be placed anywhere

along the line segment connecting the closest features on the two bodies. Although the illustration depicts the contact point as placed at the midpoint of this line segment, this selection is arbitrary. Whether the contact point is located on the first body, on the second body, or midway between the two bodies, no formula is "correct" while the bodies are separated and every formula yields the same result when the bodies are touching.

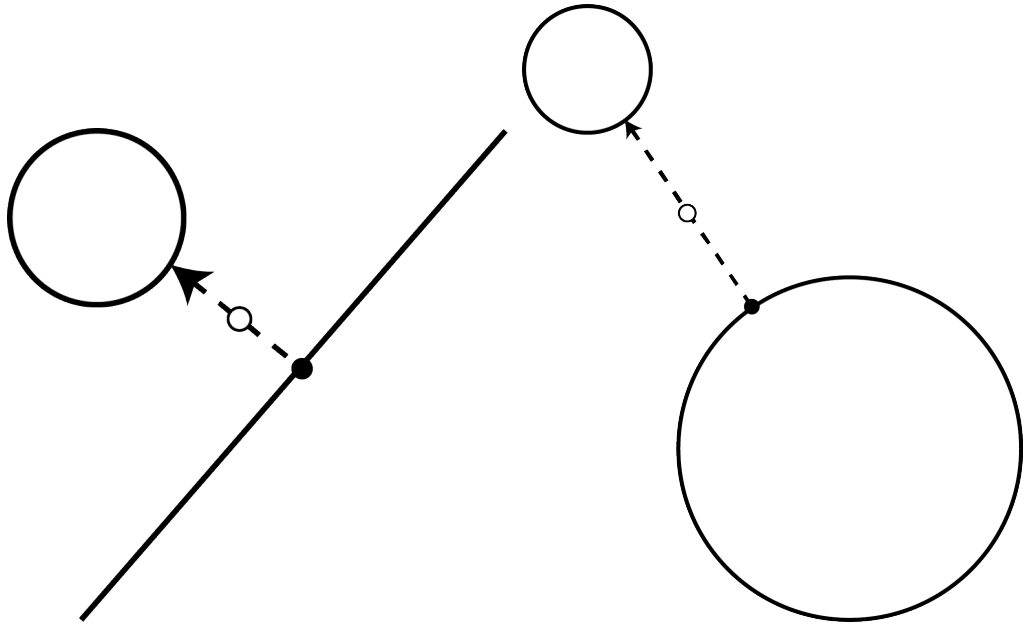


Figure 51: ***A robot's actuators are liable to be loaded while an impact occurs if contact constraints are introduced late (after the bodies have already contacted), as described in Figure 49; contact constraints may be introduced early (on the control loop*** **before** ***bodies contact) when the bodies are disjoint. This figure depicts the process of selecting points of contact and surface normals for such disjoint bodies with spherical/half-space (left) and spherical/spherical geometries (right). Closest points on the objects are connected by dotted line segments. Surface normals are depicted with an arrow. Contact points are drawn as white circles with black outlines.***

7.3 Inverse dynamics with no-slip constraints

Some contexts where inverse dynamics may be used (biomechanics, legged locomotion) may assume absence of slip (see, *e.g.*, Righetti, et al., 2011; Zhao, et al., 2014). This section describes an inverse dynamics approach that computes reaction forces from contact using the non-impacting rigid contact model with no-slip constraints. Using no-slip constraints results in a symmetric, positive semidefinite LCP. Such problems are equivalent to convex QPs by duality theory in optimization (see Cottle et al., 1992), which implies polynomial time solvability. Convexity also permits

mitigating indeterminate contact configurations, as will be seen in Section 7.3.6. This formulation inverts the rigid contact problem in a practical sense and is derived from first principles.

Two algorithms are presented in this section: Algorithm 3 ensures the no-slip constraints on the contact model are non-singular and thus guarantees that the inverse dynamics problem with contact is invertible; Algorithm 4 presents a method of mitigating torque chatter from indeterminate contact (for contexts of inverse dynamics based control) by *warm-starting* (Nocedal & Wright, 2006) the solution for the LCP solver with the last solution.

7.3.1 Normal contact constraints The equation below extends Equation 44 to multiple points of contact (via the relationship $\dot{\boldsymbol{\phi}} = \mathbf{N}\boldsymbol{v}$), where $\mathbf{N} \in \mathbb{R}^n$ is the matrix of generalized wrenches along the contact normals (see Appendix A):

$$-\frac{\bar{\boldsymbol{\phi}}}{\Delta t} \leq \bar{\mathbf{N}}^{+}\boldsymbol{v} \perp f_N \geq \mathbf{0} \tag{68}$$

Because $\boldsymbol{\phi}$ is clearly time-dependent and the control loop executes at discrete points in time, $\mathbf{N}$ must be treated as constant over a time interval. $\bar{\mathbf{N}}$ indicates that points of contact are drawn from the current configuration of the environment and multi-body. Analogous to time-stepping approaches for rigid body simulations with rigid contact, *all possible points of contact between rigid bodies over the interval* t_0 *and* t_f *can be incorporated into* $\mathbf{N}$ *as well*: as in time stepping approaches for simulation, it may not be necessary to apply forces at all of these points (the approaches implicitly can treat unnecessary points of contact as inactive, though additional computation will be necessary). Stewart (1998) showed that such an approach will converge to the solution of the continuous time dynamics as $\Delta t = (t_f - t_0) \to 0$. Given a sufficiently high control rate, Δt will be small and errors from assuming constant $\mathbf{N}$ over this interval should become negligible.

7.3.2 Discretized rigid body dynamics equation The discretized version of Equation 1, now separating contact forces into normal ($\bar{\mathbf{N}}$) and tangential wrenches ($\bar{\mathbf{S}}$ and $\bar{\mathbf{T}}$ are matrices of

generalized wrenches along the first and second contact tangent directions for all contacts) is:

$$\mathbf{M}({}^{+}\boldsymbol{v} - {}^{-}\boldsymbol{v}) = \mathbf{N}^{\mathsf{T}}\boldsymbol{f}_N + \mathbf{S}^{\mathsf{T}}\boldsymbol{f}_S + \mathbf{T}^{\mathsf{T}}\boldsymbol{f}_T - \mathbf{P}^{\mathsf{T}}\boldsymbol{\tau} + \Delta t\, {}^{-}\boldsymbol{f}_{\text{ext}} \tag{69}$$

Treating inverse dynamics at the velocity level is necessary to avoid the inconsistent configurations that can arise in the rigid contact model when forces are required to be non-impulsive (Stewart, 2000a, as also noted in Section 7.1.4). As noted above, Stewart has shown that for sufficiently small Δt, ${}^{+}\boldsymbol{v}$ converges to the solution of the continuous time dynamics and contact equations (1998).

7.3.3 Inverse dynamics constraint The inverse dynamics constraint is used to specify the desired velocities only at actuated joints:

$$\mathbf{P}\,{}^{+}\boldsymbol{v} = \dot{\boldsymbol{q}}_{\text{des}} \tag{70}$$

Desired velocities $\dot{\boldsymbol{q}}_{\text{des}}$ are calculated as:

$$\dot{\boldsymbol{q}}_{\text{des}} \equiv \dot{\boldsymbol{q}} + \Delta t \ddot{\boldsymbol{q}}_{\text{des}} \tag{71}$$

7.3.4 No-slip (infinite friction) constraints Utilizing the first-order discretization (revisit Section 7.1.4 to see why this is necessary), preventing tangential slip at a contact is accomplished by using the constraints:

$$\mathbf{S}\,{}^{+}\boldsymbol{v} = \mathbf{0} \tag{72}$$

$$\mathbf{T}\,{}^{+}\boldsymbol{v} = \mathbf{0} \tag{73}$$

These constraints indicate that the velocity in the tangent plane is zero at time t_f; the matrix representation was found to be more convenient for expression as quadratic programs and linear

complementarity problems than:

$$v_{s_i}{}^{+} = v_{t_i}{}^{+} = 0 \qquad \text{for } i = 1, \ldots, n,$$

i.e., the notation used in Section 7.1.4. All presented equations are compiled below:

Complementarity-based inverse dynamics without slip

non−interpenetration, compressive force, and normal complementarity constraints:

$$\mathbf{0} \leq \boldsymbol{f}_N \perp \mathbf{N}{}^{+}\boldsymbol{v} \geq -\frac{{}^{-}\boldsymbol{\phi}}{\Delta t}$$

no−slip constraints:

$$\mathbf{S}{}^{+}\boldsymbol{v} = \mathbf{0}$$

$$\mathbf{T}{}^{+}\boldsymbol{v} = \mathbf{0}$$

inverse dynamics:

$$\mathbf{P}{}^{+}\boldsymbol{v} = \dot{\boldsymbol{q}}_{\text{des}}$$

first−order dynamics:

$${}^{+}\boldsymbol{v} = {}^{-}\boldsymbol{v} + \mathbf{M}^{-1}(\mathbf{N}^{\mathsf{T}}\boldsymbol{f}_N + \mathbf{S}^{\mathsf{T}}\boldsymbol{f}_S + \mathbf{T}^{\mathsf{T}}\boldsymbol{f}_T - \mathbf{P}^{\mathsf{T}}\boldsymbol{\tau} + \Delta t \boldsymbol{f}_{ext})$$

Combining Equations 68–70, 72, and 73 into a *mixed linear complementarity problem* (MLCP, see Appendix 7.1.2) yields:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{P}^{\mathsf{T}} & -\mathbf{S}^{\mathsf{T}} & -\mathbf{T}^{\mathsf{T}} & -\mathbf{N}^{\mathsf{T}} \\ \mathbf{P} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{+}\boldsymbol{v} \\ \boldsymbol{\tau} \\ \boldsymbol{f}_S \\ \boldsymbol{f}_T \\ \boldsymbol{f}_N \end{bmatrix} + \begin{bmatrix} \boldsymbol{\kappa} \\ -\dot{\boldsymbol{q}}_{\text{des}} \\ \mathbf{0} \\ \mathbf{0} \\ \frac{{}^{-}\boldsymbol{\phi}}{\Delta t} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{w}_N \end{bmatrix} \tag{74}$$

$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{w}_N \geq \mathbf{0}, \boldsymbol{f}_N^{\mathsf{T}}\boldsymbol{w}_N = 0 \tag{75}$$

where $\boldsymbol{\kappa} \triangleq -\Delta t \boldsymbol{f}_{\text{ext}} - \mathbf{M}\bar{\boldsymbol{v}}$. The MLCP block matrices are defined in the form of Equations 31–153 from Appendix 7.1.2 and draw from Equations 74 and 75 to yield:

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{M} & -\mathbf{P}^{\mathsf{T}} & -\mathbf{S}^{\mathsf{T}} & -\mathbf{T}^{\mathsf{T}} \\ \mathbf{P} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \qquad \mathbf{C} \equiv \begin{bmatrix} -\mathbf{N}^{\mathsf{T}} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{D} \equiv -\mathbf{C}^{\mathsf{T}} \qquad \mathbf{B} \equiv \mathbf{0}$$

$$\boldsymbol{x} \equiv \begin{bmatrix} \overset{+}{\boldsymbol{v}} \\ \boldsymbol{\tau} \\ \boldsymbol{f}_S \\ \boldsymbol{f}_T \end{bmatrix} \qquad \boldsymbol{g} \equiv \begin{bmatrix} -\boldsymbol{\kappa} \\ \dot{\boldsymbol{q}}_{\text{des}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\boldsymbol{y} \equiv \boldsymbol{f}_N \qquad \boldsymbol{h} \equiv \frac{\bar{\boldsymbol{\phi}}}{\Delta t}$$

Applying Equations 36 and 37 (again see Appendix 7.1.2), the MLCP is transformed to a LCP $(\boldsymbol{r}, \mathbf{Q})$. Substituting in variables from the no-slip inverse dynamics model and then simplifying yields:

$$\mathbf{Q} \equiv \mathbf{C}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{C} \tag{76}$$

$$\boldsymbol{r} \equiv \frac{\bar{\boldsymbol{\phi}}}{\Delta t} + \mathbf{C}^{\mathsf{T}}\mathbf{A}^{-1}\boldsymbol{g} \tag{77}$$

The definition of matrix $\mathbf{A}$ from above may be singular, which would prevent inversion, and thereby, conversion from the MLCP to an LCP. Matrix $\mathbf{P}$ was defined as a selection matrix with full row rank, and the generalized inertia ($\mathbf{M}$) is symmetric, positive definite. If $\mathbf{S}$ and $\mathbf{T}$ have full row rank as well, or the largest subset of row blocks of $\mathbf{S}$ and $\mathbf{T}$ are identified such that full row rank is attained, $\mathbf{A}$ will be invertible as well (this can be seen by applying blockwise matrix inversion identities). Algorithm 3 performs the task of ensuring that matrix $\mathbf{A}$ is invertible. Removing

the linearly dependent constraints from the $\mathbf{A}$ matrix does not affect the solubility of the MLCP, as proved in Appendix C.

From Bhatia (2007), a matrix of $\mathbf{Q}$'s form must be non-negative definite, *i.e.*, either positive-semidefinite (PSD) or positive definite (PD). $\mathbf{Q}$ is the right product of $\mathbf{C}$ with its transpose about a symmetric PD matrix, $\mathbf{A}$. Therefore, $\mathbf{Q}$ is symmetric and either PSD or PD.

The singularity check on Lines 6 and 10 of Algorithm 3 is most quickly performed using Cholesky factorization; if the factorization is successful, the matrix is non-singular. Given that $\mathbf{M}$ is non-singular (it is symmetric, PD), the maximum size of $\mathbf{X}$ in Algorithm 3 is $m \times m$; if $\mathbf{X}$ were larger, it would be singular.

The result is that the time complexity of Algorithm 3 is dominated by Lines 6 and 10. As $\mathbf{X}$ changes by at most one row and one column per Cholesky factorization, singularity can be checked by $O(m^2)$ updates to an initial $O(m^3)$ Cholesky factorization. The overall time complexity is $O(m^3 + nm^2)$.

Algorithm 3 FIND-INDICES($\mathbf{M}, \mathbf{P}, \mathbf{S}, \mathbf{T}$), determines the row indices ($\mathcal{S}$, and $\mathcal{T}$) of $\mathbf{S}$ and $\mathbf{T}$ such that the matrix $\mathbf{A}$ (Equation 31 in Appendix 7.1.2) is non-singular.

1: $\mathcal{S} \leftarrow \emptyset$
2: $\mathcal{T} \leftarrow \emptyset$
3: **for** $i = 1, \ldots, n$ **do** ▷ n is the number of contacts
4: $\quad \mathcal{S}^* \leftarrow \mathcal{S} \cup \{i\}$
5: $\quad$ Set $\mathbf{X} \leftarrow \begin{bmatrix} \mathbf{P}^\mathsf{T} & \mathbf{S}_{\mathcal{S}^*}^\mathsf{T} & \mathbf{T}_{\mathcal{T}}^\mathsf{T} \end{bmatrix}$
6: $\quad$ **if** $\mathbf{X}^\mathsf{T}\mathbf{M}^{-1}\mathbf{X}$ not singular **then**
7: $\quad\quad \mathcal{S} \leftarrow \mathcal{S}^*$
8: $\quad \mathcal{T}^* \leftarrow \mathcal{T} \cup \{i\}$
9: $\quad$ Set $\mathbf{X} \leftarrow \begin{bmatrix} \mathbf{P}^\mathsf{T} & \mathbf{S}_{\mathcal{S}}^\mathsf{T} & \mathbf{T}_{\mathcal{T}^*}^\mathsf{T} \end{bmatrix}$
10: $\quad$ **if** $\mathbf{X}^\mathsf{T}\mathbf{M}^{-1}\mathbf{X}$ not singular **then**
11: $\quad\quad \mathcal{T} \leftarrow \mathcal{T}^*$
12: **return** $\{\mathcal{S}, \mathcal{T}\}$

7.3.5 Retrieving the inverse dynamics forces Once the contact forces have been determined, one solves Equations 69 and 70 for $\{{}^{+}\boldsymbol{v}, \boldsymbol{\tau}\}$, thereby obtaining the inverse dynamics forces. While the LCP is solvable, it is possible that the desired accelerations are inconsistent. As an example,

consider a legged robot standing on a ground plane without slip (such a case is similar to, but not identical to infinite friction, as noted in Section 7.1.4), and attempting to splay its legs outward while remaining in contact with the ground. Such cases can be readily identified by verifying that $\mathbf{N}^{+}\boldsymbol{v} \geq -\frac{\bar{\boldsymbol{\phi}}}{\Delta t}$. If this constraint is not met, consistent desired accelerations can be determined *without re-solving the LCP*. For example, one could determine accelerations that deviate minimally from the desired accelerations by solving a quadratic program:

$$\underset{^{+}\boldsymbol{v},\boldsymbol{\tau}}{\text{minimize}} \; ||\mathbf{P}^{+}\boldsymbol{v} - \dot{\boldsymbol{q}}_{\text{des}}|| \tag{78}$$

$$\text{subject to: } \mathbf{N}^{+}\boldsymbol{v} \geq -\frac{\bar{\boldsymbol{\phi}}}{\Delta t} \tag{79}$$

$$\mathbf{S}^{+}\boldsymbol{v} = \mathbf{0} \tag{80}$$

$$\mathbf{T}^{+}\boldsymbol{v} = \mathbf{0} \tag{81}$$

$$\mathbf{M}^{+}\boldsymbol{v} = \mathbf{M}^{-}\boldsymbol{v} + \mathbf{N}^{\mathsf{T}}\boldsymbol{f}_N + \mathbf{S}^{\mathsf{T}}\boldsymbol{f}_S + \mathbf{T}^{\mathsf{T}}\boldsymbol{f}_T + \mathbf{P}^{\mathsf{T}}\boldsymbol{\tau} + \Delta t \boldsymbol{f}_{\text{ext}} \tag{82}$$

This QP is always feasible: $\boldsymbol{\tau} = \mathbf{0}$ ensures that

$$\mathbf{N}^{+}\boldsymbol{v} \geq -\frac{\bar{\boldsymbol{\phi}}}{\Delta t} \tag{83}$$

$$\mathbf{S}^{+}\boldsymbol{v} = \mathbf{0} \tag{84}$$

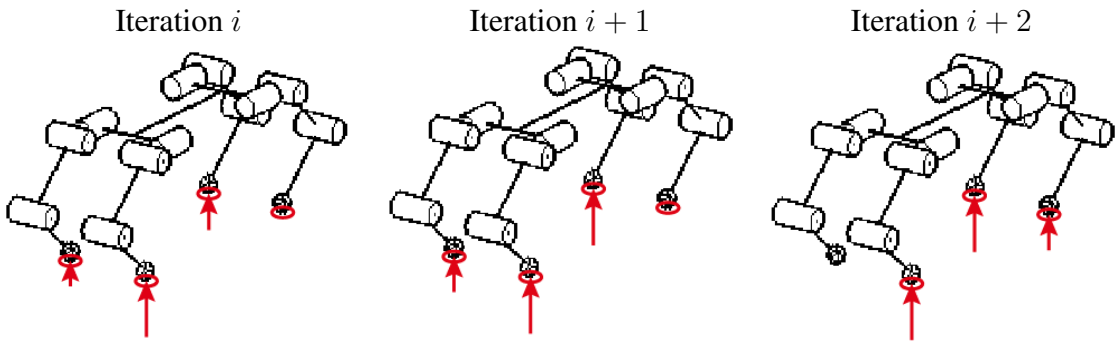
$$\mathbf{T}^{+}\boldsymbol{v} = \mathbf{0} \tag{85}$$

7.3.6 Indeterminacy mitigation A pivoting LCP solver (see Appendix B) was warm started to bias the solver toward applying forces at the same points of contact (see Figure 51)—tracking points of contact using the rigid body equations of motion—as were active on the previous inverse dynamics call (see Algorithm 4). Kuindersma et al. (2014) also use warm starting to solve a different QP resulting from contact force prediction. Although Kuindersma et al. use warm starting to generally speed the solution process, warm starting was used in this chapter to address indeter-

minacy in the rigid contact model (the inverse dynamics implementation presented in this section benefits from the increased computational speed as well).

Using warm starting, Algorithm 4 will find a solution that predicts contact forces applied at the exact same points on the last iteration *assuming that such a solution exists*. Such solutions do not exist (1) when the numbers and relative samples from the contact manifold change or (2) as contacts transition from active ($\dot{\phi}_i(\boldsymbol{x}, \boldsymbol{v}) \leq 0$) to inactive ($\dot{\phi}_i(\boldsymbol{x}, \boldsymbol{v}) > 0$), or vice versa. Case (2) implies immediate precedence or subsequence of case (1), which means that discontinuities in actuator torques will occur for at most two control loop iterations around a contact change (one discontinuity will generally occur due to the contact change itself).

Warm-Starting Example



Left ("cold start"): with four active contacts, the pivoting solver chooses three arbitrary *non-basic* indices (in $\overline{\beta}$, see Appendix B) to solve the LCP and then returns the solution. The solution applies the majority of upward force to two feet and applies a smaller amount of force to the third.
Center ("warm start"): With four active contacts, the pivoting solver chooses the *same* three non-basic indices as the last solution to attempt to solve the LCP. The warm-started solution will distribute upward forces similarly to the last solution, tending to provide consecutive solves with continuity over time.
Right ("cold start"): one foot of the robot has broken contact with the ground; there are now three active contacts. The solver returns a new solution, applying the majority of normal force to two legs, and applying a small amount of force to the third.

7.3.7 Scaling inverse dynamics runtime linearly in number of contacts The multi-body's number of generalized coordinates (m) are expected to remain constant. The number of contact points, n, depends on the multi-body's link geometries, the environment, and whether the inverse dynamics approach should anticipate *potential* contacts in $[t_0, t_f]$ (as discussed in Section 7.3.1). This section describes a method to solve inverse dynamics problems with simultaneous contact force

computation that scales linearly with additional contacts. This method will be applicable to all inverse dynamics approaches presented in this chapter except that described in Section 7.4: that problem results in a copositive LCP (Cottle et al., 1992) that the algorithm described in this section cannot generally solve.

To this point in the chapter presentation, time complexity has been dominated by the $O(n^3)$ expected time solution to the LCPs. However, a system with m degrees-of-freedom requires no more than m positive force magnitudes applied along the contact normals to satisfy the constraints for the no-slip contact model. Proof is provided in Appendix D. The following describes how that proof can be leveraged to generally decrease the expected time complexity.

Modified PPM I Algorithm This section describes a modification to the Principal Pivoting Method I (Cottle et al., 1992) (PPM) for solving LCPs (see description of this algorithm in Appendix B) that leverages the proof in Appendix D to attain expected $O(m^3+nm^2)$ time complexity. A brief explanation of the mechanics of pivoting algorithms is provided in Appendix B; the common notation of β as the set of *basic variables* and $\overline{\beta}$ as the set of *non-basic variables* is used.

The PPM requires few modifications toward the purpose of this chapter. These modifications are presented in Algorithm 4. First, the full matrix $\mathbf{N}\mathbf{M}^{-1}\mathbf{N}^{\mathsf{T}}$ is never constructed, because the construction is unnecessary and would require $O(n^3)$ time. Instead, Line 11 of the algorithm constructs a maximum $m \times m$ system; thus, that operation requires only $O(m^3)$ operations. Similarly, Lines 12 and 13 also leverage the proof from Appendix D to compute $\boldsymbol{w}^{\dagger}$ and $\boldsymbol{a}^{\dagger}$ efficiently (though these operations do not affect the asymptotic time complexity). Expecting that the number of iterations for a pivoting algorithm is $O(n)$ in the size of the input (Cottle et al., 1992) and assuming that each iteration requires at most two pivot operations (each rank-1 update operation to a matrix factorization will exhibit $O(m^2)$ time complexity), the asymptotic complexity of the modified PPM I algorithm is $O(m^3 + m^2n)$. The termination conditions for the algorithm are not affected by the modifications.

The reader should note that Baraff has proven that LCPs of the form $(\mathbf{H}\boldsymbol{w}, \mathbf{H}\mathbf{Q}^{-1}\mathbf{H}^{\mathsf{T}})$, where $\mathbf{H} \in \mathbb{R}^{p\times q}, \mathbf{Q} \in \mathbb{R}^{q\times q}, \mathbf{w} \in \mathbb{R}^{q}$, and $\mathbf{Q}$ is symmetric, PD are solvable. Thus, the inverse dynamics

Algorithm 4 $\{\boldsymbol{z}, \boldsymbol{w}, \overline{\mathcal{B}}\} = \text{PPM}(\boldsymbol{N}, \boldsymbol{M}, \boldsymbol{f}^*, \boldsymbol{z}^-)$ Solves the LCP $(\mathbf{N}\mathbf{M}^{-1}\boldsymbol{f}^*, \mathbf{N}\mathbf{M}^{-1}\mathbf{N}^\mathsf{T})$ resulting from convex, rigid contact models (the no-slip model and the complementarity-free model with Coulomb friction) . $\overline{\mathcal{B}}^*$ are the set of non-basic indices returned from the last call to PPM.

$n \leftarrow \text{rows}(\mathbf{N})$
$\boldsymbol{r} \leftarrow \mathbf{N} \cdot \boldsymbol{f}^*$
$i \leftarrow \arg\min_i r_i$ ▷ Check for trivial solution
if $r_i \geq 0$ **then**
    return $\{\mathbf{0}, \boldsymbol{r}\}$
$\overline{\mathcal{B}} \leftarrow \overline{\mathcal{B}}^*$
if $\overline{\mathcal{B}} = \emptyset$ **then**
    $\overline{\mathcal{B}} \leftarrow \{i\}$ ▷ Establish initial nonbasic indices
$\mathcal{B} \leftarrow \{1, \ldots, n\} - \overline{\mathcal{B}}$ ▷ Establish basic indices
while *true* **do**
    $\mathbf{A} \leftarrow \mathbf{N}_{\overline{\mathcal{B}}} \cdot \mathbf{M}^{-1} \cdot \mathbf{N}_{\overline{\mathcal{B}}}{}^\mathsf{T}$
    $\boldsymbol{b} \leftarrow \mathbf{N}_{\overline{\mathcal{B}}} \cdot \boldsymbol{f}^*$
    $\boldsymbol{z}^\dagger \leftarrow \mathbf{A}^{-1} \cdot -\boldsymbol{b}$ ▷ Compute $\boldsymbol{z}$ non-basic components
    $\boldsymbol{a}^\dagger \leftarrow \mathbf{M}^{-1} \cdot \mathbf{N}_{\overline{\mathcal{B}}}{}^\mathsf{T} \boldsymbol{z}^\dagger + \boldsymbol{f}^*$
    $\boldsymbol{w}^\dagger \leftarrow \mathbf{N} \cdot \boldsymbol{a}^\dagger$
    $i \leftarrow \arg\min_i w_i^\dagger$ ▷ Search for index to move into non-basic set
    if $w_i^\dagger \geq 0$ **then**
        $j \leftarrow \arg\min_i \boldsymbol{z}_i^\dagger$ ▷ No index to move into the non-basic set; search for index to move into the basic set
        if $z_j^\dagger < 0$ **then**
            $k \leftarrow \overline{\mathcal{B}}(j)$
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{k\}$ ▷ Move index k into basic set
            $\overline{\mathcal{B}} \leftarrow \overline{\mathcal{B}} - \{k\}$
            continue
        else
            $\boldsymbol{z} \leftarrow \mathbf{0}$
            $\boldsymbol{z}_{\overline{\mathcal{B}}} \leftarrow \boldsymbol{z}^\dagger$
            $\boldsymbol{w} \leftarrow \mathbf{0}$
            $\boldsymbol{w}_{\mathcal{B}} \leftarrow \boldsymbol{w}^\dagger$
            return $\{\boldsymbol{z}, \boldsymbol{w}\}$
    else
        $\overline{\mathcal{B}} \leftarrow \overline{\mathcal{B}} \cup \{i\}$ ▷ Move index i into non-basic set
        $\mathcal{B} \leftarrow \mathcal{B} - \{i\}$
        $j \leftarrow \arg\min_i \boldsymbol{z}_i^\dagger$ ▷ Try to find an index to move into the basic set
        if $z_j^\dagger < 0$ **then**
            $k \leftarrow \overline{\mathcal{B}}(j)$
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{k\}$ ▷ Move index k into basic set
            $\overline{\mathcal{B}} \leftarrow \overline{\mathcal{B}} - \{k\}$

model will always possess a solution (1994).

7.4 Inverse dynamics with Coulomb friction

Though control with the absence of slip may facilitate grip and traction, the assumption is often not the case in reality. Foot and manipulator geometries, and planned trajectories must be specially planned to prevent sliding contact, and assuming sticking contact may lead to disastrous results (see discussion on experimental results in Section 7.7.2). Implementations of controllers that limit actuator forces to keep contact forces within the bounds of a friction constraint have been suggested to reduce the occurrence of unintentional slip in walking robots (Righetti et al., 2013). These methods also limit the reachable space of accelerative trajectories that the robot may follow, as all movements yielding sliding contact would be infeasible. The model presented in this section permits sliding contact. This section formulates inverse dynamics using the computational model of rigid contact with Coulomb friction developed by Stewart & Trinkle (1996) and Anitescu & Potra (1997); the equations in this section closely follow those in Anitescu & Potra (1997).

7.4.1 Coulomb friction constraints Still utilizing the first-order discretization of the rigid body dynamics (Equation 69), the linearized Coulomb friction constraints are reproduced from Anitescu & Potra (1997) without further explanation (identical except for slight notational differences):

$$\mathbf{0} \leq \mathbf{E}\boldsymbol{\lambda} + {}^{-}\mathbf{F}^{+}\boldsymbol{v} \perp \boldsymbol{f}_F \geq \mathbf{0} \tag{86}$$

$$\mathbf{0} \leq \boldsymbol{\mu} f_N - \mathbf{E}^{\mathsf{T}}\boldsymbol{f}_F \perp \boldsymbol{\lambda} \geq \mathbf{0} \tag{87}$$

where $\mathbf{E} \in \mathbb{R}^{n \times nk}$ (k is the number of edges in the polygonal approximation to the friction cone) retains its definition from Anitescu & Potra (1997) as a sparse selection matrix containing blocks of "ones" vectors, $\boldsymbol{\mu} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with elements corresponding to the coefficients of friction at the n contacts, $\boldsymbol{\lambda}$ is a variable roughly equivalent to magnitude of tangent velocities after contact forces are applied, and $\mathbf{F} \in \mathbb{R}^{nk \times m}$ (equivalent to $\mathbf{D}$ in Anitescu & Potra, 1997) is the matrix of wrenches of frictional forces at k tangents to each contact point. If the friction cone is

approximated by a pyramid (an assumption made throughout the remainder of the chapter), then:

$$\mathbf{F} \equiv \begin{bmatrix} \mathbf{S}^\mathsf{T} & -\mathbf{S}^\mathsf{T} & \mathbf{T}^\mathsf{T} & -\mathbf{T}^\mathsf{T} \end{bmatrix}^\mathsf{T}$$
$$\boldsymbol{f}_F \equiv \begin{bmatrix} \boldsymbol{f}_S^{+\mathsf{T}} & \boldsymbol{f}_S^{-\mathsf{T}} & \boldsymbol{f}_T^{+\mathsf{T}} & \boldsymbol{f}_T^{-\mathsf{T}} \end{bmatrix}^\mathsf{T}$$

where $\boldsymbol{f}_S = \boldsymbol{f}_S^+ - \boldsymbol{f}_S^-$ and $\boldsymbol{f}_T = \boldsymbol{f}_T^+ - \boldsymbol{f}_T^-$. Given these substitutions, the contact model with inverse dynamics becomes:

Complementarity-based inverse dynamics

non−interpenetration, compressive force, and normal complementarity constraints:

$$\mathbf{0} \leq \boldsymbol{f}_N \perp \mathbf{N}\,{}^{+}\boldsymbol{v} \geq -\frac{{}^{-}\boldsymbol{\phi}}{\Delta t} \tag{88}$$

Coulomb friction constraints:

$$\mathbf{0} \leq \lambda \boldsymbol{e} + \mathbf{F}\,{}^{+}\boldsymbol{v} \perp \boldsymbol{f}_F \geq \mathbf{0} \tag{89}$$

$$\mathbf{0} \leq \mu f_N - \boldsymbol{e}^\mathsf{T} \boldsymbol{f}_F \perp \boldsymbol{\lambda} \geq \mathbf{0} \tag{90}$$

inverse dynamics:

$$\mathbf{P}\,{}^{+}\boldsymbol{v} = \dot{\boldsymbol{q}}_{\text{des}} \tag{91}$$

first−order dynamics:

$${}^{+}\boldsymbol{v} = {}^{-}\boldsymbol{v} + \mathbf{M}^{-1}(\mathbf{N}^\mathsf{T}\boldsymbol{f}_N + \mathbf{F}^\mathsf{T}\boldsymbol{f}_F + \Delta t \boldsymbol{f}_{ext} - \mathbf{P}^\mathsf{T}\boldsymbol{\tau}) \tag{92}$$

7.4.2 Resulting MLCP Combining Equations 68–70 and 86–87 results in the MLCP:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{P}^\mathsf{T} & -\mathbf{N}^\mathsf{T} & -\mathbf{F}^\mathsf{T} & \mathbf{0} \\ \mathbf{P} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{F} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E} \\ \mathbf{0} & \mathbf{0} & \mu & -\mathbf{E}^\mathsf{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{+}\boldsymbol{v} \\ \boldsymbol{\tau} \\ \boldsymbol{f}_N \\ \boldsymbol{f}_F \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{\kappa} \\ -\dot{\boldsymbol{q}}_{\text{des}} \\ \frac{{}^{-}\boldsymbol{\phi}}{\Delta t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{w}_N \\ \boldsymbol{w}_F \\ \boldsymbol{w}_\lambda \end{bmatrix} \tag{93}$$

$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{w}_N \geq \mathbf{0}, \boldsymbol{f}_N^{\mathsf{T}} \boldsymbol{w}_N = 0 \tag{94}$$

$$\boldsymbol{f}_F \geq \mathbf{0}, \boldsymbol{w}_F \geq \mathbf{0}, \boldsymbol{f}_F^{\mathsf{T}} \boldsymbol{w}_F = 0 \tag{95}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{w}_\lambda \geq \mathbf{0}, \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{w}_\lambda = 0 \tag{96}$$

Vectors $\boldsymbol{w}_N$ and $\boldsymbol{w}_F$ correspond to the normal and tangential velocities after impulsive forces have been applied.

Transformation to LCP and proof of solution existence The MLCP can be transformed to a LCP as described by Cottle et al. (1992) by solving for the unconstrained variables ${}^{+}\boldsymbol{v}$ and $\boldsymbol{\tau}$. This transformation is possible because the matrix:

$$\mathbf{X} \equiv \begin{bmatrix} \mathbf{M} & \mathbf{P}^{\mathsf{T}} \\ \mathbf{P} & \mathbf{0} \end{bmatrix} \tag{97}$$

is non-singular. Proof comes from blockwise invertibility of this matrix, which requires only invertibility of $\mathbf{M}$ (guaranteed because generalized inertia matrices are positive definite) and $\mathbf{P}\mathbf{M}^{-1}\mathbf{P}^{\mathsf{T}}$. This latter matrix selects exactly those rows and columns corresponding to the joint space inertia matrix (Featherstone, 1987), which is also positive definite. After eliminating the unconstrained variables ${}^{+}\boldsymbol{v}$ and $\boldsymbol{\tau}$, the following LCP results:

$$\begin{bmatrix} \mathbf{N}\mathbf{M}^{-1}\mathbf{N}^{\mathsf{T}} & \mathbf{N}\mathbf{M}^{-1}\mathbf{F}^{\mathsf{T}} & \mathbf{E} \\ \mathbf{F}\mathbf{M}^{-1}\mathbf{N}^{\mathsf{T}} & \mathbf{F}\mathbf{M}^{-1}\mathbf{F}^{\mathsf{T}} & \mathbf{0} \\ -\mathbf{E}^{\mathsf{T}} & \boldsymbol{\mu} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_N \\ \boldsymbol{f}_F \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} \frac{\bar{}\phi}{\Delta t} - \mathbf{N}\mathbf{M}^{-1}\boldsymbol{\kappa} \\ -\mathbf{F}\mathbf{M}^{-1}\boldsymbol{\kappa} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_N \\ \boldsymbol{w}_F \\ \boldsymbol{w}_\lambda \end{bmatrix} \tag{98}$$

$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{w}_N \geq \mathbf{0}, \boldsymbol{f}_N^{\mathsf{T}} \boldsymbol{w}_N = 0 \tag{99}$$

$$\boldsymbol{f}_F \geq \mathbf{0}, \boldsymbol{w}_F \geq \mathbf{0}, \boldsymbol{f}_F^{\mathsf{T}} \boldsymbol{w}_F = 0 \tag{100}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{w}_\lambda \geq \mathbf{0}, \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{w}_\lambda = 0 \tag{101}$$

The discussion in Stewart & Trinkle (1996) can be used to show that this LCP matrix is *copositive* (see Cottle, et al., 1992, Definition 3.8.1), since for any vector $\boldsymbol{z} = \begin{bmatrix} \boldsymbol{f}_N{}^{\mathsf{T}} & \boldsymbol{f}_F{}^{\mathsf{T}} & \boldsymbol{\lambda}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \geq \mathbf{0}$,

$$\begin{bmatrix} \boldsymbol{f}_N \\ \boldsymbol{f}_F \\ \boldsymbol{\lambda} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathbf{N}\mathbf{M}^{-1}\mathbf{N}^{\mathsf{T}} & \mathbf{N}\mathbf{M}^{-1}\mathbf{F}^{\mathsf{T}} & \mathbf{E} \\ \mathbf{F}\mathbf{M}^{-1}\mathbf{N}^{\mathsf{T}} & \mathbf{F}\mathbf{M}^{-1}\mathbf{F}^{\mathsf{T}} & \mathbf{0} \\ \boldsymbol{\mu} & -\mathbf{E}^{\mathsf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_N \\ \boldsymbol{f}_F \\ \boldsymbol{\lambda} \end{bmatrix} =$$

$$(\mathbf{N}^{\mathsf{T}} \boldsymbol{f}_N + \mathbf{F}^{\mathsf{T}} \boldsymbol{f}_N)^{\mathsf{T}} \mathbf{M}^{-1} (\mathbf{N}^{\mathsf{T}} \boldsymbol{f}_N + \mathbf{F}^{\mathsf{T}} \boldsymbol{f}_N) + \boldsymbol{f}_N{}^{\mathsf{T}} \boldsymbol{\mu} \boldsymbol{\lambda} \geq 0 \tag{102}$$

because $\mathbf{M}^{-1}$ is positive definite and $\boldsymbol{\mu}$ is a diagonal matrix with non-negative elements. The transformation from the MLCP to the LCP yields $(k+2)n$ LCP variables (the per-contact allocation is: one for the normal contact magnitude, k for the frictional force components, and one for an element of $\boldsymbol{\lambda}$) and at most $2m$ unconstrained variables.

As noted by Anitescu & Potra (1997), Lemke's Algorithm can provably solve such copositive LCPs (Cottle et al., 1992) if precautions are taken to prevent cycling through indices. After solving the LCP, joint torques can be retrieved exactly as in Section 7.3.5. *Thus, this section has shown—using elementary extensions to the work in Anitescu & Potra (1997)—that a right inverse of the non-impacting rigid contact model exists* (as first broached in Section 7.0.1). Additionally, the expected running time of Lemke's Algorithm is cubic in the number of variables, so this inverse can be computed in expected polynomial time.

7.4.3 Contact indeterminacy Though the approach presented in this section produces a solution to the inverse dynamics problem with simultaneous contact force computation, the approach can

converge to a vastly different, but equally valid solution at each controller iteration. However, unlike the no-slip model, it is unclear how to bias the solution to the LCP, because a means for warm starting Lemke's Algorithm is currently unknown (previous experience confirms the common wisdom that using the basis corresponding to the last solution usually leads to excessive pivoting).

Generating a torque profile that would evolve without generating torque chatter requires checking all possible solutions of the LCP if this approach is to be used for robot control. Generating all solutions requires a linear system solve for each combination of basic and non-basic indices among all problem variables. Enumerating all possible solutions yields exponential time complexity, the same as the worst case complexity of Lemke's Algorithm (Lemke, 1965). After all solutions have been enumerated, torque chatter would be eliminated by using the solution that differs minimally from the last solution. Algorithm 5 presents this approach.

Algorithm 5 $\{\boldsymbol{x}, \epsilon\}$ $=$MINDIFF$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \boldsymbol{g}, \boldsymbol{h}, \overline{\mathcal{B}}, \boldsymbol{x}_0, n)$ Computes the solution to the LCP $(\boldsymbol{h} - \mathbf{D}\mathbf{A}^{-1}\boldsymbol{g}, \mathbf{B} - \mathbf{D}\mathbf{A}^{-1}\mathbf{C})$ that is closest (by Euclidean norm) to vector $\boldsymbol{x}_0$ using a recursive approach. n is always initialized as rows($\mathbf{B}$).

if $n > 0$ **then**
  $\{\boldsymbol{x}_1, \epsilon_1\}$ $=$MINDIFF$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \boldsymbol{g}, \boldsymbol{h}, \overline{\mathcal{B}}, \boldsymbol{x}_0, n-1)$
  $\overline{\mathcal{B}} \leftarrow \{\overline{\mathcal{B}}, n\}$ ▷ Establish nonbasic indices
  $\{\boldsymbol{x}_2, \epsilon_2\}$ $=$MINDIFF$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \boldsymbol{g}, \boldsymbol{h}, \overline{\mathcal{B}}, \boldsymbol{x}_0, n-1)$
  if $\epsilon_1 < \epsilon_2$ **then return** $\{\boldsymbol{x}_1, \epsilon_1\}$
  else **return** $\{\boldsymbol{x}_2, \epsilon_2\}$
else
  $\{\boldsymbol{z}, \boldsymbol{w}\} = \mathrm{LCP}(\boldsymbol{h}_{nb} - \mathbf{D}_{nb}\mathbf{A}^{-1}\boldsymbol{g}, \mathbf{B}_{nb} - \mathbf{D}_{nb}\mathbf{A}^{-1}\mathbf{C}_{nb})$
  $\boldsymbol{x} \leftarrow \mathbf{A}^{-1}(\mathbf{C}_{nb}\boldsymbol{z}_{nb} + \boldsymbol{g})$
  return $\{\boldsymbol{x}, \|\boldsymbol{x} - \boldsymbol{x}_0\|\}$

The fact that all solutions to the problem can be enumerated in exponential time proves that solving the problem is at worst NP-hard, though following an enumerative approach is not practical.

7.5 Convex inverse dynamics without normal complementarity

This section describes an approach for inverse dynamics that mitigates indeterminacy in rigid contact using the impact model described in Section 7.1.4. The approach is almost identical to the "standard" rigid contact model described in Section 7.1.4, but for the absence of the normal complementarity constraint.

The approach works by determining contact and actual forces in a first step and then solving within the nullspace of the objective function (Equation 61) such that joint forces are minimized. The resulting problem is strictly convex, and thus torques are continuous in time (and more likely safe for a robot to apply) if the underlying dynamics are smooth. This latter assumption is violated only when a discontinuity occurs from one control loop iteration to the next, as it would when contact is broken, bodies newly come into contact, the contact surface changes, or contact between two bodies switches between slipping and sticking.

7.5.1 Two-stage vs. single-stage approaches Torque chatter due to contact indeterminacy can be avoided by ensuring that contact forces do not cycle rapidly between points of contact under potentially indeterminate contact configurations across successive controller iterations. Zapolsky & Drumwright (2014) eliminate torque chatter using a QP stage that searches over the optimal set of contact forces (using a convex relaxation to the rigid contact model) for forces that minimize the ℓ_2-norm of joint torques.

An alternative, single-stage approach is described by Todorov (2014), who regularizes the quadratic objective matrix to attain the requisite strict convexity. Another single-stage approach (which is tested in Section 7.6) uses the warm starting-based solution technique described in Section 7.3 to mitigate contact indeterminacy.

Single-stage approaches are expected to generally run faster. However, the two-stage approach described below confers the following advantages over single-stage approaches: (1) no regularization factor need be chosen—there has yet to be a physical interpretation behind regularization factors, and computing a minimal regularization factor (via, e.g., singular value decomposition) would be computationally expensive; and (2) the two-stage approach allows the particular solution to be selected using an arbitrary objective criterion—minimizing actuator torques is particularly relevant for robotics applications. Although two stage approaches are slower, performance suitably fast for real-time control loops on quadrupedal robots has been demonstrated in previous work (Zapolsky & Drumwright, 2014). The two-stage approach is presented without further comment, as the reader can realize the single-stage approach readily by regularizing the Hessian matrix in

the quadratic program.

7.5.2 Computing inverse dynamics and contact forces simultaneously (Stage I) For simplicity of presentation, it is assumed that the number of edges in the approximation of the friction cone for each contact is four; in other words, a friction pyramid will be used in place of a cone. The inverse dynamics problem is formulated as follows:

As discussed in Section 7.1.4 the contact model always has a solution (*i.e.*, the QP is always feasible) and that the contact forces will not do positive work (Drumwright & Shell, 2010). The addition of the inverse dynamics constraint (Equation 108) will not change this result—the frictionless version of this QP is identical to an LCP of the form that Baraff has proven solvable (see Section 7.3.7), which means that the QP is feasible. As in the inverse dynamics approach in Section 7.4, the first order approximation to acceleration avoids inconsistent configurations that can occur in rigid contact with Coulomb friction. The worst-case time complexity of solving this convex model is polynomial in the number of contact features (Boyd & Vandenberghe, 2004). High frequency control loops limit n to approximately four contacts given present hardware and using fast active-set methods.

Complementarity-free inverse dynamics: Stage I

$$\underset{\boldsymbol{f}_N, \boldsymbol{f}_F, {}^{+}\boldsymbol{v}, \boldsymbol{\tau}}{\text{minimize}} \quad \text{dissipate kinetic energy maximally:} \quad \frac{1}{2} {}^{+}\boldsymbol{v}^{\mathsf{T}} \mathbf{M} {}^{+}\boldsymbol{v} \tag{103}$$

subject to:

non−interpenetration constraint:

$$\mathbf{N} {}^{+}\boldsymbol{v} \geq -\frac{{}^{-}\boldsymbol{\phi}}{\Delta t} \tag{104}$$

variable non−negativity (for formulation convenience):

$$\boldsymbol{f}_N \geq \mathbf{0}, \quad \boldsymbol{f}_F \geq \mathbf{0} \tag{105}$$

Coulomb friction:

$$\mu f_{N_i} \geq \mathbf{1}^{\mathsf{T}} \boldsymbol{f}_{F_i} \tag{106}$$

first−order velocity relationship:

$${}^{+}\boldsymbol{v} = {}^{-}\boldsymbol{v} + \mathbf{M}^{-1} \left(\mathbf{N}^{\mathsf{T}} \boldsymbol{f}_N + \mathbf{F}^{\mathsf{T}} \boldsymbol{f}_F + \Delta t \boldsymbol{f}_{ext} - \mathbf{P}^{\mathsf{T}} \boldsymbol{\tau} \right) \tag{107}$$

inverse dynamics:

$$\mathbf{P} {}^{+}\boldsymbol{v} = \dot{\boldsymbol{q}}_{\text{des}} \tag{108}$$

Removing equality constraints The optimization in this section is a convex quadratic program with inequality and equality constraints. The equality constraints are removed through substitution. This reduces the size of the optimization problem; removing linear equality constraints also eliminates significant variables if transforming the QP to a LCP via optimization duality theory (Cottle et al., 1992).[4]

[4] such a transformation is used in this chapter, permitting the application of the `LEMKE` solver (Fackler & Miranda, 2011), which is freely available, numerically robust (using Tikhonov regularization), and relatively fast.

The resulting QP takes the form:

$$\underset{\boldsymbol{f}_N, \boldsymbol{f}_F}{\text{minimize}} \begin{bmatrix} \boldsymbol{f}_N \\ \boldsymbol{f}_F \end{bmatrix}^\mathsf{T} \left(\begin{bmatrix} \mathbf{N}\mathbf{X}^{-1}\mathbf{N}^\mathsf{T} & \mathbf{N}\mathbf{X}^{-1}\mathbf{F}^\mathsf{T} \\ \mathbf{F}\mathbf{X}^{-1}\mathbf{N}^\mathsf{T} & \mathbf{F}\mathbf{X}^{-1}\mathbf{F}^\mathsf{T} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_N \\ \boldsymbol{f}_F \end{bmatrix} + \begin{bmatrix} -\mathbf{N}\boldsymbol{\kappa} \\ -\mathbf{F}\boldsymbol{\kappa} \end{bmatrix} \right) \tag{109}$$

$$\text{subject to: } \begin{bmatrix} \mathbf{N}\mathbf{X}^{-1}\mathbf{N}^\mathsf{T} & \mathbf{N}\mathbf{X}^{-1}\mathbf{F}^\mathsf{T} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_N \\ \boldsymbol{f}_F \end{bmatrix} - \mathbf{N}\boldsymbol{\kappa} \geq \mathbf{0} \tag{110}$$

$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{f}_F \geq \mathbf{0} \tag{111}$$

$$\mu f_{N_i} \geq \mathbf{1}^\mathsf{T} \boldsymbol{f}_{F_i} \tag{112}$$

Once $\boldsymbol{f}_N$ and $\boldsymbol{f}_F$ have been determined, the inverse dynamics forces are computed using:

$$\begin{bmatrix} {}^{+}\boldsymbol{v} \\ \boldsymbol{\tau} \end{bmatrix} = \mathbf{X}^{-1} \begin{bmatrix} -\boldsymbol{\kappa} + \mathbf{N}^\mathsf{T}\boldsymbol{f}_N + \mathbf{F}^\mathsf{T}\boldsymbol{f}_F \\ \dot{\boldsymbol{q}}_{\text{des}} \end{bmatrix} \tag{113}$$

As in Section 7.3.5, consistency in the desired accelerations can be verified and modified without re-solving the QP if found to be inconsistent.

Minimizing floating point computations Because inverse dynamics may be used within real-time control loops, this section describes an approach that can minimize floating point computations over the formulation described above.

Assume that the joint forces $\boldsymbol{f}_{ID}$ necessary to solve the inverse dynamics problem are first solved for under no contact constraints. The new velocity ${}^{+}\boldsymbol{v}$ is now defined as:

$${}^{+}\boldsymbol{v} = {}^{-}\boldsymbol{v} + \mathbf{M}^{-1} \left(\mathbf{N}^\mathsf{T}\boldsymbol{f}_N + \mathbf{F}^\mathsf{T}\boldsymbol{f}_F + \Delta t \boldsymbol{f}_{ext} + \begin{bmatrix} \mathbf{0} \\ \Delta t(\boldsymbol{f}_{ID} + \boldsymbol{x}) \end{bmatrix} \right) \tag{114}$$

where $\boldsymbol{x}$ is defined to be the actuator forces that are added to $\boldsymbol{f}_{ID}$ to counteract contact forces. To simplify the derivations for this inverse dynamic formulation, the following vectors and matrices are defined:

$$\mathbf{R} \equiv \begin{bmatrix} \mathbf{N}^\mathsf{T} & \mathbf{F}^\mathsf{T} \end{bmatrix} \tag{115}$$

$$\boldsymbol{z} \equiv \begin{bmatrix} \boldsymbol{f}_N & \boldsymbol{f}_F \end{bmatrix}^\mathsf{T} \tag{116}$$

$$\mathbf{M} \equiv \begin{matrix} \\ nb \\ nq \end{matrix} \begin{matrix} \begin{matrix} nb & nq \end{matrix} \\ \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\mathsf{T} & \mathbf{C} \end{bmatrix} \end{matrix} \tag{117}$$

$$\mathbf{M}^{-1} \equiv \begin{matrix} \\ nb \\ nq \end{matrix} \begin{matrix} \begin{matrix} nb & nq \end{matrix} \\ \begin{bmatrix} \mathbf{D} & \mathbf{E} \\ \mathbf{E}^\mathsf{T} & \mathbf{G} \end{bmatrix} \end{matrix} \tag{118}$$

$$\boldsymbol{j} \equiv \boldsymbol{v}_b + \begin{bmatrix} \mathbf{D} & \mathbf{E} \end{bmatrix} \left(\Delta t \boldsymbol{f}_{ext} + \begin{bmatrix} \mathbf{0} \\ \Delta t \boldsymbol{f}_{ID} \end{bmatrix} \right) \tag{119}$$

$$\boldsymbol{k} \equiv \boldsymbol{v}_q + \begin{bmatrix} \mathbf{E}^\mathsf{T} & \mathbf{G} \end{bmatrix} \left(\Delta t \boldsymbol{f}_{ext} + \begin{bmatrix} \mathbf{0} \\ \Delta t \boldsymbol{f}_{ID} \end{bmatrix} \right) \tag{120}$$

$$\boldsymbol{f}_{ID} \equiv \mathbf{C}\ddot{\boldsymbol{q}} - \boldsymbol{f}_{ext,nq} \tag{121}$$

Where nq is the total joint degrees of freedom of the robot, and nb is the total base degrees of freedom for the robot ($nb = 6$ for floating bases).

The components of ${}^{+}\boldsymbol{v}$ are then defined as follows:

$${}^{+}\boldsymbol{v}_b \equiv \boldsymbol{j} + \begin{bmatrix} \mathbf{D} & \mathbf{E} \end{bmatrix} \left(\mathbf{R}\boldsymbol{z} + \begin{bmatrix} \mathbf{0} \\ \Delta t \boldsymbol{x} \end{bmatrix} \right) \tag{122}$$

$${}^{+}\boldsymbol{v}_q \equiv \boldsymbol{k} + \begin{bmatrix} \mathbf{E}^\mathsf{T} & \mathbf{G} \end{bmatrix} \left(\mathbf{R}\boldsymbol{z} + \begin{bmatrix} \mathbf{0} \\ \Delta t \boldsymbol{x} \end{bmatrix} \right) = \dot{\boldsymbol{q}}_{\text{des}} \tag{123}$$

Solving for $\boldsymbol{x}$ from the latter equation:

$$\boldsymbol{x} = \frac{\mathbf{G}^{-1}\left({}^{+}\boldsymbol{v}_q - \boldsymbol{k} - \begin{bmatrix}\mathbf{E}^{\mathsf{T}} & \mathbf{G}\end{bmatrix}\mathbf{R}\boldsymbol{z}\right)}{\Delta t} \tag{124}$$

Equation 124 indicates that once contact forces are computed, determining the actuator forces for inverse dynamics requires solving only a linear equation. Substituting the solution for $\boldsymbol{x}$ into Eqn. 122 yields:

$${}^{+}\boldsymbol{v}_b = \boldsymbol{j} + \begin{bmatrix}\mathbf{D} & \mathbf{E}\end{bmatrix}\mathbf{R}\boldsymbol{z} + \mathbf{E}\mathbf{G}^{-1}\left({}^{+}\boldsymbol{v}_q - \boldsymbol{k} - \begin{bmatrix}\mathbf{E}^{\mathsf{T}} & \mathbf{G}\end{bmatrix}\mathbf{R}\boldsymbol{z}\right) \tag{125}$$

To simplify further derivation, a new matrix and a new vector are defined:

$$\mathbf{Z} \equiv \left(\begin{bmatrix}\mathbf{D} & \mathbf{E}\end{bmatrix} - \mathbf{E}\mathbf{G}^{-1}\begin{bmatrix}\mathbf{E}^{\mathsf{T}} & \mathbf{G}\end{bmatrix}\right)\mathbf{R} \tag{126}$$

$$\boldsymbol{p} \equiv \boldsymbol{j} + \mathbf{E}\mathbf{G}^{-1}({}^{+}\boldsymbol{v}_q - \boldsymbol{k}) \tag{127}$$

Now, ${}^{+}\boldsymbol{v}_b$ can be defined simply, and solely in terms of $\boldsymbol{z}$, as:

$${}^{+}\boldsymbol{v}_b \equiv \mathbf{Z}\boldsymbol{z} + \boldsymbol{p} \tag{128}$$

The objective function (Eqn. 103) can now be represented in block form as:

$$f(.) \equiv \frac{1}{2}\begin{bmatrix}{}^{+}\boldsymbol{v}_b \\ {}^{+}\boldsymbol{v}_q\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}\mathbf{A} & \mathbf{B} \\ \mathbf{B}^{\mathsf{T}} & \mathbf{C}\end{bmatrix}\begin{bmatrix}{}^{+}\boldsymbol{v}_b \\ {}^{+}\boldsymbol{v}_q\end{bmatrix} \tag{129}$$

which is identical to:

$$f(.) \equiv \frac{1}{2}{}^{+}\boldsymbol{v}_b{}^{\mathsf{T}}\mathbf{A}\,{}^{+}\boldsymbol{v}_b + {}^{+}\boldsymbol{v}_b\mathbf{B}^{\mathsf{T}}\,{}^{+}\boldsymbol{v}_q + \frac{1}{2}{}^{+}\boldsymbol{v}_q{}^{\mathsf{T}}\mathbf{C}\,{}^{+}\boldsymbol{v}_q \tag{130}$$

As minimizing $f(.)$ with regard to $\boldsymbol{z}$ does not depend on the last term of the above equation, that term is ignored hereafter. Expanding remaining terms using Equation 122, the new objective

function is:

$$f(.) \equiv \frac{1}{2}\boldsymbol{z}^{\mathsf{T}}\mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}\boldsymbol{z} + \boldsymbol{z}^{\mathsf{T}}\mathbf{Z}^{\mathsf{T}}\mathbf{A}\boldsymbol{p} + \boldsymbol{z}^{\mathsf{T}}\mathbf{Z}^{\mathsf{T}}\mathbf{B}^{+}\boldsymbol{v}_q \tag{131}$$

$$\equiv \frac{1}{2}\boldsymbol{z}^{\mathsf{T}}\mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}\boldsymbol{z} + \boldsymbol{z}^{\mathsf{T}}\left(\mathbf{Z}^{\mathsf{T}}\mathbf{A}\boldsymbol{p} + \mathbf{Z}^{\mathsf{T}}\mathbf{B}^{+}\boldsymbol{v}_q\right) \tag{132}$$

subject to the following constraints:

$$\mathbf{N}^{\mathsf{T}}\begin{bmatrix}\mathbf{Z}\boldsymbol{z} + \boldsymbol{p} \\ \boldsymbol{v}_q^*\end{bmatrix} \geq -\frac{\bar{\boldsymbol{\phi}}}{\Delta t} \tag{133}$$

$$\boldsymbol{f}_{N,i} \geq 0 \qquad (\text{for } i = 1 \ldots n) \tag{134}$$

$$\mu\boldsymbol{f}_{N,i} \geq \boldsymbol{c}_{S,i} + \boldsymbol{c}_{T,i} \tag{135}$$

Symmetry and positive semi-definiteness of the QP follows from symmetry and positive definiteness of $\mathbf{A}$. Once the solution to this QP is determined, the actuator forces $\boldsymbol{x} + \boldsymbol{f}_{ID}$ determined via inverse dynamics can be recovered.

Floating point operation counts Operation counts for matrix-vector arithmetic and numerical linear algebra are taken from Hunger (2007).

Before simplifications: Floating point operations (*flops*) necessary to setup the *Stage I* model as presented initially sum to 77,729 flops, substituting: $m = 18, nq = 16, n = 4, k = 4$.

operation	*flops*
$LDL^{\mathsf{T}}(\mathbf{X})$	$\frac{m^3}{3} + m^2(\text{nq}) + m^2 + m(\text{nq})^2 + 2m(\text{nq}) - \frac{7m}{3} + \frac{(\text{nq})^3}{3} + (\text{nq})^2 - \frac{7(\text{nq})}{3} + 1$
$\mathbf{X}^{-1}\mathbf{N}^{\mathsf{T}}$	$m + 2m^2n + (nq) + 4mn(nq) + 2n(nq)^2$
$\mathbf{X}^{-1}\mathbf{F}^{\mathsf{T}}$	$m + 2km^2n + (nq) + 4kmn(nq) + 2kn(nq)^2$
$\mathbf{N}\mathbf{X}^{-1}\mathbf{N}^{\mathsf{T}}$	$2mn^2 - mn$
$\mathbf{F}\mathbf{X}^{-1}\mathbf{N}^{\mathsf{T}}$	$2mn^2k - mn$
$\mathbf{F}\mathbf{X}^{-1}\mathbf{F}^{\mathsf{T}}$	$2mnk^2 - mnk$
κ	$2m^2 - m$
$\mathbf{X}^{-1}\kappa$	$2(m + (nq))^2$
$\mathbf{N}\mathbf{X}^{-1}\kappa$	$mn - m$
$\mathbf{F}\mathbf{X}^{-1}\kappa$	$mnk - m$
$\boldsymbol{\tau}$	$2(m + (nq))^2$

Table 10: *Floating point operations (*flops*) per task without floating point optimizations.*

After simplifications: Floating point operations necessary to setup the *Stage I* model after modified to reduce computational costs sum to 73,163 flops when substituting: $m = 18, nq = 16, n = 4, k = 4, nb = m - nq$, a total of 6.24% reduction in computation. When substituting: $m = 18, nq = 12, n = 4, k = 4, nb = m - nq$, 102,457 flops were observed for this new method and 62,109 flops before simplification. Thus, a calculation of the total number of floating point operations should be performed to determine whether the floating point simplification is actually more efficient for a particular robot.

operation	*flops*
$\left(LL^{\mathsf{T}}(\mathbf{M})\right)^{-1}$	$\frac{2}{3}m^3 + \frac{1}{2}m^2 + \frac{5}{6}m$
$LL^{\mathsf{T}}(\mathbf{G})$	$\frac{1}{3}(nq)^3 + \frac{1}{2}(nq)^2 + \frac{1}{6}nq$
$\mathbf{Z}$	$nb\,m + nq\,n(nk+1)(2m-1) + nb\,m(2nq-1) + 2nq^2m$
$\boldsymbol{p}$	$2nb\,nq + 2nq^2 + 3nq + 2nb + 2m + 2m\,nq$
$\mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}$	$2nb^2(n(nk+1)) - nb\,(n(nk+1)) + 2nb\,(n(nk+1))^2 - (n(nk+1))^2$
$\mathbf{Z}^{\mathsf{T}}\mathbf{A}\boldsymbol{p}$	$(n(nk+1) + nb)(2nb-1)$
$\mathbf{Z}^{\mathsf{T}}\mathbf{B}\boldsymbol{v}_q^*$	$(n(nk+1) + nq)(2nq-1)$
$\mathbf{N}^{\mathsf{T}}\mathbf{Z}$	$n^2(nk+1)(2nb-1)$
$\mathbf{N}^{\mathsf{T}}\left[\begin{smallmatrix}\boldsymbol{p}\\ \boldsymbol{v}_q^*\end{smallmatrix}\right]$	$n(2m-1)$

Table 11: *Floating point operations (*flops*) with floating point optimizations.*

Recomputing Inverse Dynamics to Stabilize Actuator Torques (Stage II) In the case that the matrix $\mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}$ is singular, the contact model is only convex rather than strictly convex (Boyd &

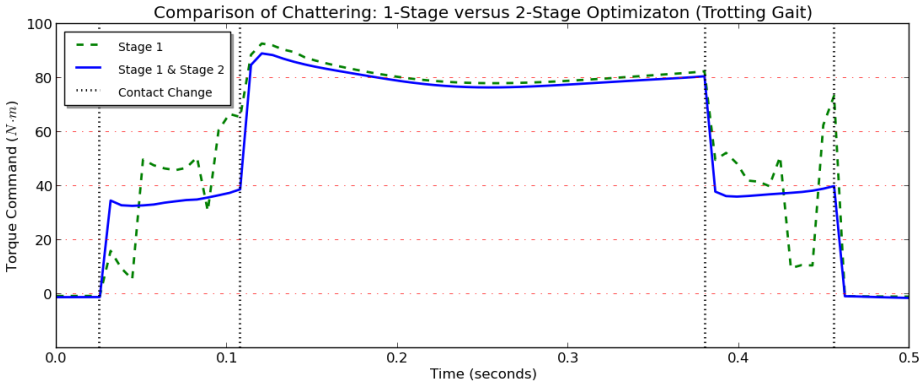


Figure 52: ***Plot of torque chatter while controlling with inverse dynamics using an indeterminate contact model (Stage 1) versus the smooth torque profile produced by a determinate contact model (Stage 1 & Stage 2).***

Vandenberghe, 2004). Where a strictly convex system has just one optimal solution, the convex problem may have multiple equally optimal solutions. Conceptually, contact forces that predict that two legs, three legs, or four legs support a walking robot are all equally valid. A method is thus needed to optimize within the contact model's solution space while favoring solutions that predict contact forces at all contacting links (and thus preventing the rapid torque cycling between arbitrary optimal solutions). As Section 7.1.5 noted, defining a manner by which actuator torques evolve over time, or selecting a preferred distribution of contact forces may remedy the issues resulting from indeterminacy. One such method would select—from the space of optimal solutions to the indeterminate contact problem—the one that minimizes the ℓ_2-norm of joint torques. If the solution that was computed in the last section is denoted as $\boldsymbol{z}_0$, the following optimization problem will yield the desired result:

Complementarity-free inverse dynamics: Stage II

find minimum norm motor torques:

$$\underset{\boldsymbol{f}_N, \boldsymbol{f}_F}{\text{minimize}} \; \frac{1}{2}\boldsymbol{\tau}^\mathsf{T}\boldsymbol{\tau} \tag{136}$$

subject to: Equations 104–108

maintain Stage I objective:

$$\frac{1}{2}\overset{+}{\boldsymbol{v}}^\mathsf{T}\mathbf{M}\overset{+}{\boldsymbol{v}} \leq f(\boldsymbol{z}_0) \tag{137}$$

The method described in Section 7.5.2 was referred to as *Stage I* and the optimization problem

above *Stage II*. Constraining for a quadratic objective function (Equation 103) with a quadratic inequality constraint yields a QCQP that may not be solvable sufficiently quickly for high frequency control loops. This section now presents how to use the nullspace of $\mathbf{Z}^\mathsf{T}\mathbf{A}\mathbf{Z}$ to perform this second optimization *without explicitly considering the quadratic inequality constraint*; thus, a QP problem formulation is retained. Assume that the matrix $\mathbf{W}$ gives the nullspace of $\mathbf{Z}^\mathsf{T}\mathbf{A}\mathbf{Z}$. The vector of contact forces will now be given as $\boldsymbol{z} + \mathbf{W}\boldsymbol{w}$, where $\boldsymbol{w}$ will be the optimization vector.

The kinetic energy from applying the contact impulses is:

$$\begin{aligned}\epsilon_2 &= \frac{1}{2}(\boldsymbol{z} + \mathbf{W}\boldsymbol{w})^\mathsf{T}\mathbf{Z}^\mathsf{T}\mathbf{A}\mathbf{Z}(\boldsymbol{z} + \mathbf{W}\boldsymbol{w}) + (\boldsymbol{z} + \mathbf{W}\boldsymbol{w})^\mathsf{T}(\mathbf{Z}^\mathsf{T}\mathbf{A}\boldsymbol{p} + \mathbf{Z}^\mathsf{T}\mathbf{B}^+\boldsymbol{v}_q) \\ &= \frac{1}{2}\boldsymbol{z}^\mathsf{T}\mathbf{Z}^\mathsf{T}\mathbf{A}\mathbf{Z}\boldsymbol{z} + \boldsymbol{z}^\mathsf{T}(\mathbf{Z}^\mathsf{T}\mathbf{A}\boldsymbol{p} + \mathbf{Z}^\mathsf{T}\mathbf{B}^+\boldsymbol{v}_q) + \boldsymbol{w}^\mathsf{T}\mathbf{W}^\mathsf{T}(\mathbf{Z}^\mathsf{T}\mathbf{A}\boldsymbol{p} + \mathbf{Z}^\mathsf{T}\mathbf{B}^+\boldsymbol{v}_q)\end{aligned}$$

The terms $\frac{1}{2}\boldsymbol{w}^\mathsf{T}\mathbf{W}^\mathsf{T}\mathbf{Z}^\mathsf{T}\mathbf{A}\mathbf{Z}\mathbf{W}\boldsymbol{w}$ and $\boldsymbol{z}\mathbf{Z}^\mathsf{T}\mathbf{A}\mathbf{Z}\mathbf{W}\boldsymbol{w}$ are not included above because both are zero: $\mathbf{W}$ is in the nullspace of $\mathbf{Z}^\mathsf{T}\mathbf{A}\mathbf{Z}$. The energy dissipated in the second stage. ϵ_2, should be equal to the energy dissipated in the first stage, ϵ_1. Thus $\epsilon_2 - \epsilon_1 = 0$ is the value of interest. Algebra yields:

$$\boldsymbol{w}^\mathsf{T}\mathbf{W}^\mathsf{T}(\mathbf{Z}^\mathsf{T}\mathbf{A}\boldsymbol{p} + \mathbf{Z}^\mathsf{T}\mathbf{B}^+\boldsymbol{v}_q) = 0 \tag{138}$$

The ℓ_2-norm of joint torques is minimize with respect to contact forces by first defining $\boldsymbol{y}$ as:

$$\boldsymbol{y} \equiv \frac{\mathbf{G}^{-1}\left(^+\boldsymbol{v}_q - \boldsymbol{k} - \begin{bmatrix}\mathbf{E}^\mathsf{T} & \mathbf{G}\end{bmatrix}\mathbf{R}(\boldsymbol{z} + \mathbf{W}\boldsymbol{w})\right)}{\Delta t} \tag{139}$$

The resulting objective is:

$$g(\boldsymbol{w}) \equiv \frac{1}{2}\boldsymbol{y}^\mathsf{T}\boldsymbol{y} \tag{140}$$

From this, the following optimization problem arises:

$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y} \tag{141}$$

$$\text{subject to: } (\boldsymbol{p}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}} + {}^{+}\boldsymbol{v}_q{}^{\mathsf{T}}\mathbf{B}^{\mathsf{T}})\mathbf{Z}\mathbf{W}\boldsymbol{w} = \mathbf{0} \tag{142}$$

$$\mathbf{N}^{\mathsf{T}}\begin{bmatrix}\mathbf{Z}(\boldsymbol{z} + \mathbf{W}\boldsymbol{w}) + \boldsymbol{p} \\ {}^{+}\boldsymbol{v}_q\end{bmatrix} \geq -\frac{{}^{-}\boldsymbol{\phi}}{\Delta t} \tag{143}$$

$$(\boldsymbol{z} + \mathbf{W}\boldsymbol{w})_i \geq \mathbf{0}, \quad (\text{for } i = 1 \dots 5n) \tag{144}$$

$$\mu_i(\boldsymbol{z} + \mathbf{W}\boldsymbol{w})_i \geq \mathbf{X}_{S_i}(\boldsymbol{z} + \mathbf{W}\boldsymbol{w})_{S_i} + \mathbf{X}_{T_i}(\boldsymbol{z} + \mathbf{W}\boldsymbol{w})_{T_i} \qquad (\text{for } i = 1 \dots n) \tag{145}$$

Equation 144 constrains the contact force vector to only positive values, accounting for 5 positive directions to which force can be applied at the contact manifold $(\hat{n}, \hat{s}, -\hat{s}, \hat{t}, -\hat{t})$.[5] A proof that $\mathbf{Z} \cdot \ker(\mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}) = \mathbf{0}$ (Zapolsky & Drumwright, 2014) is used to render $n+1$ (Equations 142 and 143) of $7n+1$ linear constraints (Equations 142–145) unnecessary.

Finally, expanding and simplifying Equation 140 (removing terms that do not contain $\boldsymbol{w}$, scaling by Δt^2), and using the identity $\mathbf{U} \equiv \begin{bmatrix}\mathbf{E}^{\mathsf{T}} & \mathbf{G}\end{bmatrix}\mathbf{R}$ yields:

$$\begin{aligned} g(\boldsymbol{w}) \equiv & \frac{1}{2}\boldsymbol{w}^{\mathsf{T}}\mathbf{W}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}\mathbf{G}^{-1^{\mathsf{T}}}\mathbf{G}^{-1}\mathbf{U}\mathbf{W}\boldsymbol{w} \\ & + \boldsymbol{z}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}\mathbf{G}^{-1^{\mathsf{T}}}\mathbf{G}^{-1}\mathbf{U}\mathbf{W}\boldsymbol{w} - {}^{+}\boldsymbol{v}_q{}^{\mathsf{T}}\mathbf{G}^{-1^{\mathsf{T}}}\mathbf{G}^{-1}\mathbf{U}\mathbf{W}\boldsymbol{w} \\ & + \boldsymbol{k}^{\mathsf{T}}\mathbf{G}^{-1^{\mathsf{T}}}\mathbf{G}^{-1}\mathbf{U}\mathbf{W}\boldsymbol{w} \end{aligned} \tag{146}$$

Finally, actuator torques for the robot can be retrieved by calculating $\boldsymbol{y} + \boldsymbol{f}_{ID}$.

Feasibility and time complexity It should be clear that a feasible point ($\boldsymbol{w} = \mathbf{0}$) always exists for the optimization problem. The dimensionality ($n \times n$ in the number of contact points) of $\mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}$ yields a nullspace computation of $O(n^3)$ and represents one third of the Stage II running times in the experiments testing this controller. For quadrupedal robots with single point contacts,

[5] negative and positive $\hat{s}$ and $\hat{t}$ are considered because `LEMKE` requires all variables to be positive.

for example, the dimensionality of $\boldsymbol{w}$ is typically at most two, yielding fewer than $6n + 2$ total optimization variables (each linear constraint introduces six KKT dual variables for the simplest friction cone approximation). Timing results given n contacts for this virtual robot are available in Section 7.7.4.

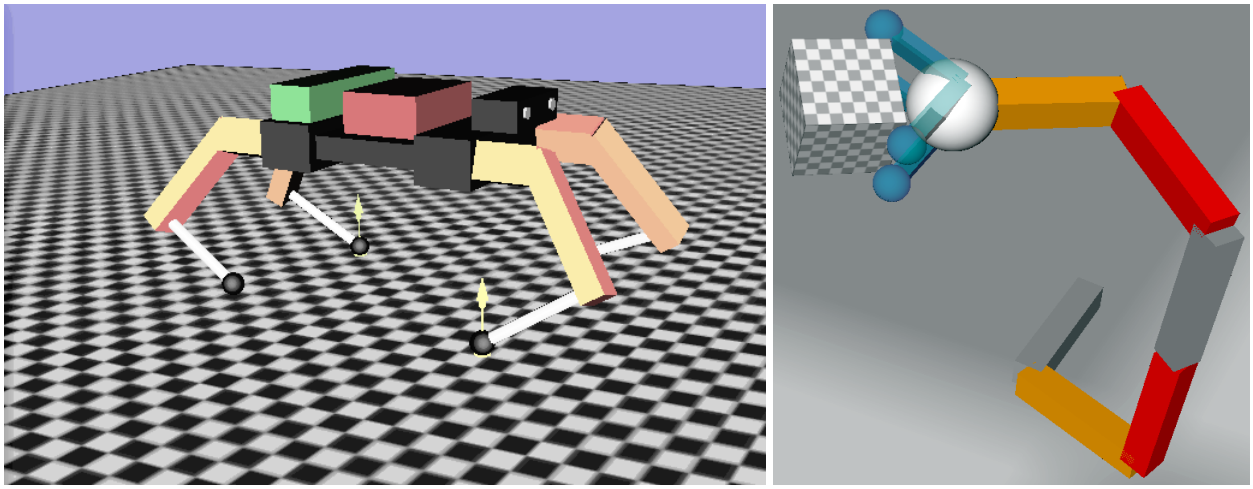
7.6 Experiments

This section assesses the inverse dynamics controllers under a range of conditions on a virtual, locomoting quadrupedal robot (depicted in Figures 53a) and a virtual manipulator grasping a box. For points of comparison, performance data for three reference controllers (depicted in Figures 54a and 54c) are also provided. These experiments also serve to illustrate that the inverse dynamics approaches function as expected.

The effects of possible modeling infidelities and sensing inaccuracies are assessed by testing locomotion performance on rigid planar terrain, rigid non-planar terrain, and on compliant planar terrain. The last of these is an example of modeling infidelity, as the compliant terrain violates the assumption of rigid contact. Sensing inaccuracies may be introduced from physical sensor noise or perception error producing, *e.g.*, erroneous friction or contact normal estimates. All code and data for experiments conducted in simulation, as well as videos of the virtual robots, are located (and can thus be reproduced) at:

`http://github.com/PositronicsLab/idyn-experiments`.

7.6.1 Platforms The performance of all controllers are evaluated on a simulated quadruped (see Figure 53a). This test platform measures 16 cm tall and has three degree of freedom legs. Its feet are modeled as spherical collision geometries, creating a a single point contact manifold with the terrain. This platform is used to assess the effectiveness of the inverse dynamics implementations with one to four points of contact. Results presented from this platform are applicable to biped locomotion as well, the only differentiating factor being the presence of a more involved planning and balance system than the one driving the quadruped.



(a) ***A quadruped robot in the*** **MOBY** ***simulator on planar terrain.***

(b) ***A fixed-base manipulator in the*** **MOBY** ***simulator grasping a box with four spherical fingertips.***

Figure 53: ***Snapshots of the simulated robotic platforms that were considered in the experiments.***

Additionally, adaptability of this approach is demonstrated on a manipulator grasping a box (see Figure 53b). The arm has seven degrees of freedom and each finger has one degree of freedom, totaling eleven actuated degrees of freedom. The finger tips have spherical collision geometries, creating a a single point contact manifold with a grasped object at each fingertip. The grasped box has six non-actuated degrees of freedom with a surface friction that is specified in each experiment.

7.6.2 Source of planned trajectories The system used to generate locomotion trajectories for quadrupedal robots is a reactive planning framework that produces a continuous velocity command with cubicly splined trajectories (see9). The locomotion framework is used to validate the inverse dynamics controller presented in this chapter because it allows us to control exclusively with accelerative joint commands which are then passed to the inverse dynamics system to produce the only joint torques supplied to the robot's actuators. The quadruped's performance in this experiment is thus a product of exclusively the accuracy of the inverse dynamics system.

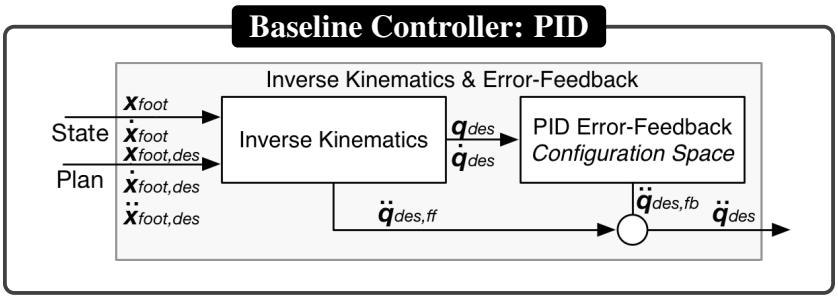
Arm trajectory planner The fixed-base manipulator is command to follow a simple sinusoidal trajectory parameterized over time. The arm oscillates through its periodic motion about three times per second. The four fingers gripping the box during the experiment are commanded to

maintain zero velocity and acceleration while gripping the box, and to close further if not contacting the grasped object.

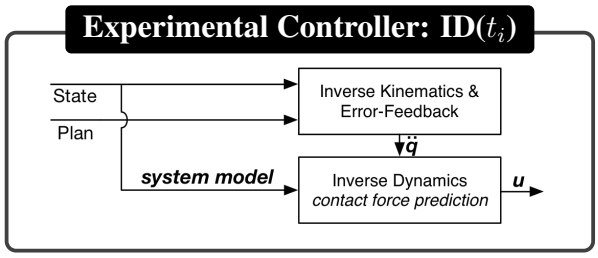
7.6.3 Evaluated controllers The same error-feedback control scheme is used in all cases for the purpose of reducing joint tracking error from drift (see baseline controller in Figure 54a). The gains used for PID control are identical between all controllers but differ between robots. The PID error feedback controller is defined in configuration-space on all robots. Balance and stabilization are handled in this trajectory planning stage, balancing the robot as it performs its task. The stabilization implementation uses an inverted pendulum model for balance, applying only virtual compressive forces along the contact normal to stabilize the robot (Sugihara & Nakamura, 2003). The error-feedback and stabilization modules also accumulate error-feedback from configuration-space errors into the vector of desired acceleration ($\ddot{q}_{des}$) input into the inverse dynamics controllers.

The new controllers presented in this chapter will hereafter be referred to as $\mathbf{ID}(t_i)_{\texttt{solver,friction}}$, where the possible solvers are: $\texttt{solver} = \{\text{QP, LCP}\}$ for QP and LCP-based optimization models, respectively; and the possible friction models are: $\texttt{friction} = \{\mu, \infty\}$ for finite Coulomb friction and no-slip models, respectively.

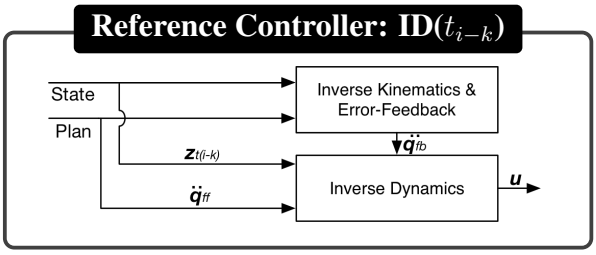
This experimental section compared the controllers implemented using the methods described in Sections 7.3, 7.4 and 7.5, see Figure 54b) against the reference controllers (Figure 54c), using finite and infinite friction coefficients to permit comparison against no-slip and Coulomb friction models, respectively. Time "t_i" in $\mathbf{ID}(t_i)$ refers to the use of contact forces predicted at the current controller time. The experimental (presented) controllers include: $\mathbf{ID}(t_i)_{LCP,\infty}$ is the *ab initio* controller from Section 7.3 that uses an LCP model, to predict contact reaction forces with no-slip constraints; $\mathbf{ID}(t_i)_{LCP,\mu}$ is the *ab initio* controller from Section 7.4 that uses an LCP model, to predict contact reaction forces with Coulomb friction; $\mathbf{ID}(t_i)_{QP,\mu}$ is the controller from Section 7.5 that uses a QP-based optimization approach for contact force prediction; $\mathbf{ID}(t_i)_{QP,\infty}$ is the same controller as $\mathbf{ID}(t_i)_{QP,\mu}$ from Section 7.5.2, but set to allow infinite frictional forces.



(a) ***Reference PID joint error-feedback controller, PD operational space error-feedback controller (quadruped only), and VIIP stabilization (quadruped only).***



(b) ***Inverse dynamics controller with predictive contact forces (this work) generates an estimate of contact forces at the current time ($\hat{z}(t)$) given contact state and internal forces.***



(c) ***Reference inverse dynamics controller using exact sensed contact forces from the k^{th} most recent contact force measurement, $z(t - k\Delta t)$***

Figure 54: ***Controllers used in experiments of this chapter.***

The reference inverse dynamics controllers use sensed contact forces; the sensed forces are the exact forces applied by the simulator to the robot on the previous simulation step (*i.e.*, there is a sensing lag of Δt on these measurements, the simulation step size). The controller using these exact sensed contact forces are denoted as **ID**(t_{i-1}). Controller "**ID**(t_{i-1})" uses the exact value of sensed forces from the immediately previous time step in simulation and represents an upper limit on the performance of using sensors to incorporate sensed contact forces into an inverse dynamics model. *In situ* implementation of contact force sensing should result in worse performance than the controller described here, as it would be subject to inaccuracies in sensing and delays of multiple controller cycles as sensor data is filtered to smooth noise; the effect of a second controller cycle delay with **ID**(t_{i-2}) is examined as well (see Figure 54c).

7.6.4 Software and simulation setup PACER runs alongside the open source simulator MOBY[6], which was used to simulate the legged locomotion scenarios used in the experiments. MOBY was arbitrarily set to simulate contact with the Stewart-Trinkle / Anitescu-Potra rigid contact models (Stewart & Trinkle, 1996; Anitescu & Potra, 1997); therefore, the contact models utilized by the simulator match those used in the reference controllers, *permitting us to compare the contact force predictions directly against those determined by the simulator*. Both simulations and controllers had access to identical data: kinematics (joint locations, link lengths, *etc.*), dynamics (generalized inertia matrices, external forces), and friction coefficients at points of contact. MOBY provides accurate time of contact calculation for impacting bodies, yielding more accurate contact point and normal information. Other simulators step past the moment of contact, and approximate contact information based on the intersecting geometries. The accurate contact information provided by MOBY allows us to test the inverse dynamics controllers under more realistic conditions: contact may break or be formed between control loops.

7.6.5 Terrain types for locomotion experiments The performance of the baseline, reference, and presented controllers are evaluated on a planar terrain. Four cases are used to encompass many typical surface properties experienced in locomotion. Sticky and slippery terrain are modeled with frictional properties corresponding to a Coulomb friction of 0.1 for low (slippery) friction and infinity for high (sticky) friction. Rigid and compliant terrain are also modeled, using rigid and penalty (spring-damper) based contact model, respectively. The compliant terrain is modeled to be relatively hard, yielding only 1 mm interpenetration of the foot into the plane while the quadruped is at rest. The contact prediction models used to implement the inverse dynamic controllers presented in this chapter all assume rigid contact; a compliant terrain will assess whether the inverse dynamics model for predictive contact is viable when the contact model of the environment does not match its internal model. Successful locomotion using such a controller on a compliant surface would indicate (to some confidence) that the controller is robust to modeling infidelities and thus more robust in an uncertain environment.

[6]Obtained from `https://github.com/PositronicsLab/Moby`

Finally, the controllers are tested on a rigid height map terrain with non-vertical surface normals and varied surface friction to assess robustness on a more natural terrain. The variability of the terrain is limited to 3 cm in height (about one fifth the height of the quadruped see Figure 55). so that the performance of the foothold planner (Chapter 9) would not bias performance results. Friction values were selected between the upper and lower limits of Coulomb friction ($\mu \sim \mathcal{U}(0.1, 1.5)$) found in various literature on legged locomotion.

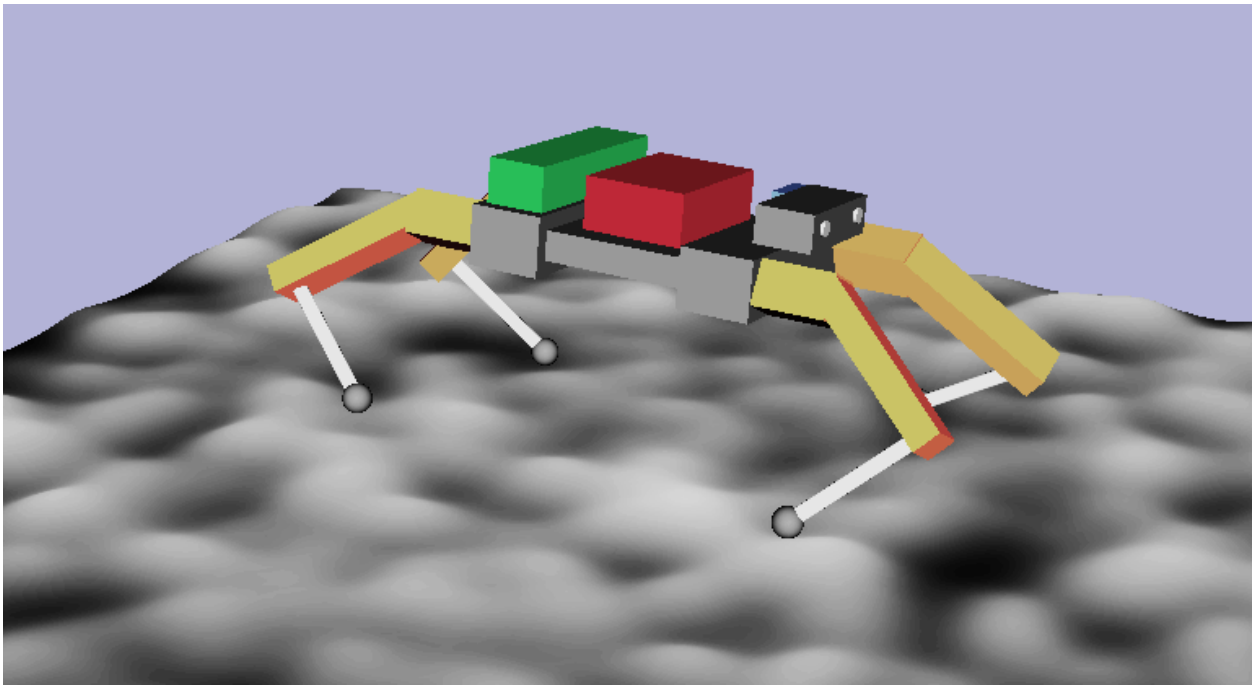


Figure 55: ***Snapshot of a quadruped robot in the* MOBY *simulator on rough terrain.***

7.6.6 Tasks The quadruped was directed to trot between a set of waypoints on a planar surface for 30 virtual seconds. The process of trotting to a waypoint and turning toward a new goal stresses some basic abilities needed to locomote successfully: (1) acceleration to and from rest; (2) turning in place, and (3) turning while moving forward. For the trotting gait a gait duration of 0.3 seconds per cycle was assigned with a duty-factor of 75% of the total gait duration, a step height of 1.5 cm, and touchdown times $\{0.25, 0.75, 0.75, 0.25\}$ for {left front, right front, left hind, right hind} feet, respectively. These values were chosen as a generally functional set of parameters for a trotting gait on a 16 cm tall quadruped. The desired forward velocity of the robot over the test

was 20 cm/s. Over the interval of the experiment, the robots are in both determinate and *possibly* indeterminate contact configurations and, as noted above, undergo numerous contact transitions. Section 7.7 shows that the controllers presented in Sections 7.4 and 7.3 are feasible for controlling a quadruped robot over a trot; contact force predictions made by all presented controllers are compared to the reaction forces generated by the simulation (Section 7.7.2); running times of all presented controllers given numerous additional contacts in Section 7.7.4 are measured as well.

The manipulator grasping a box is directed to follow a simple, sinusoidal joint trajectory. During this process the hand is susceptible to making contact transitions as the box slips from the grasp. The divergence from the desired trajectory of the box is recorded over the course of each experiment. The reader should note that the objective of this task is to accurately follow the joint trajectory—predicting joint torques and contact forces with rigid contact constraints—not to hold onto the box firmly.

7.7 Results

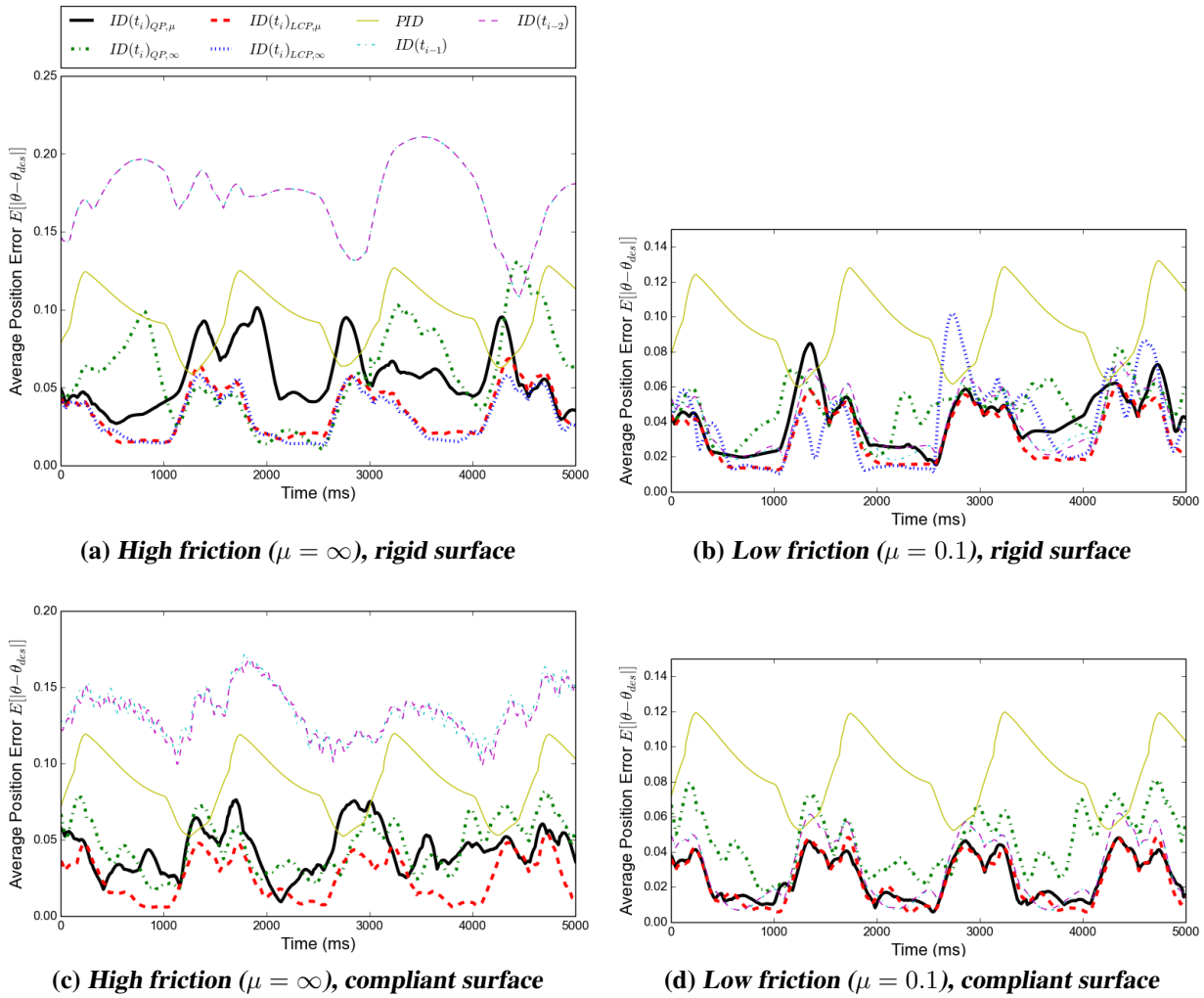
This section quantifies and plots results from the experiments in the previous section in five ways: (1) joint trajectory tracking; (2) accuracy of contact force prediction; (3) torque command smoothness; (4) center-of-mass behavior over a gait; (5) computation speed.

Trajectory tracking on planar surfaces: Tracking performance using the quadruped platform is analyzed on both rigid planar and compliant planar surfaces (see Figure 56). Joint tracking data was collected from the simulated quadruped using the baseline, reference and experimental controllers to locomote on a planar terrain with varying surface properties. Numerical results for these experiments are presented in Table 12. The experimental controllers implementing contact force prediction,**ID(**t_i**)**, either outperformed or matched the performance of the inverse dynamics formulations using sensed contact, **ID(**t_{i-1}**)** and **ID(**t_{i-2}**)**.

As expected, the baseline controller (PID) performed substantially worse than all inverse dynamics systems for positional tracking on a low friction surface. Also, only the inverse dynamics controllers that use predictive contact managed to gracefully locomote with no-slip contact.

The reference inverse dynamics controllers with sensed contact performed the worst on high friction surfaces, only serving to degrade locomotion performance from the baseline controller over the course of the experiment. The high level of performance for the PID error-feedback controller was attributed the control system absorbing some of the error introduced by poor planning; In such a case, more accurate tracking of planned trajectories might lead to worse overall locomotion stability.

Trajectory Tracking: Quadruped



(a) ***High friction*** **($\mu = \infty$)*****, rigid surface***

(b) ***Low friction*** **($\mu = 0.1$)*****, rigid surface***

(c) ***High friction*** **($\mu = \infty$)*****, compliant surface***

(d) ***Low friction*** **($\mu = 0.1$)*****, compliant surface***

Figure 56: Average position error for all joints ***($E[|\theta - \theta_{des}|]$) over time while the quadruped performs a trotting gait.***

Trajectory Tracking Error

Rigid, Low Friction		
Controller	positional error	velocity error
ID$(t_i)_{QP,\mu}$	0.0310	1.9425
ID$(t_i)_{QP,\infty}$	0.0483	2.6295
ID$(t_i)_{LCP,\mu}$	**0.0239**	1.8386
ID$(t_i)_{LCP,\infty}$	-	-
PID	0.0895	**1.5569**
ID(t_{i-1})	0.0325	1.7952
ID(t_{i-2})	0.0328	1.7830

Rigid, High Friction		
Controller	positional error	velocity error
ID$(t_i)_{QP,\mu}$	0.0486	2.2596
ID$(t_i)_{QP,\infty}$	0.0654	2.5737
ID$(t_i)_{LCP,\mu}$	**0.0259**	1.8784
ID$(t_i)_{LCP,\infty}$	0.0260	1.8950
PID	0.0916	**1.4653**
ID(t_{i-1})	0.1317	2.5585
ID(t_{i-2})	0.1316	2.5608

Compliant, Low Friction		
Controller	positional error	velocity error
ID$(t_i)_{QP,\mu}$	**0.0217**	2.0365
ID$(t_i)_{QP,\infty}$	-	-
ID$(t_i)_{LCP,\mu}$	0.0219	2.0786
ID$(t_i)_{LCP,\infty}$	-	-
PID	0.0850	**1.5845**
ID(t_{i-1})	0.0265	1.8858
ID(t_{i-2})	0.0267	1.8742

Compliant, High Friction		
Controller	positional error	velocity error
ID$(t_i)_{QP,\mu}$	0.0342	2.9360
ID$(t_i)_{QP,\infty}$	0.0446	3.9779
ID$(t_i)_{LCP,\mu}$	**0.0226**	2.1243
ID$(t_i)_{LCP,\infty}$	-	-
PID	0.0850	**1.5845**
ID(t_{i-1})	0.1270	4.4377
ID(t_{i-2})	0.1270	4.2061

Table 12: *Expected trajectory tracking error for quadrupedal locomotion (positional: mean magnitude of radian error for all joints over trajectory duration ($E[E[|\theta - \theta_{des}|]]$), velocity: mean magnitude of radians/second error for all joints over trajectory duration ($E[E[|\dot{\theta} - \dot{\theta}_{des}|]]$)) of inverse dynamics controllers (**ID(..)**) and baseline (**PID**) controller.*

Torque Chatter

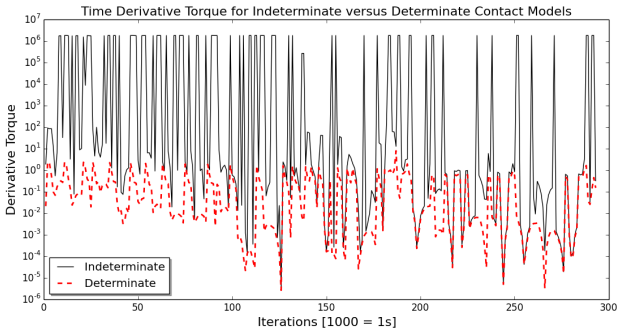


Figure 57: ***Time derivative torque when using the inverse dynamics method (ID$(t_i)_{QP,\mu}$) with means for mitigating torque chatter from indeterminate contact (red/dotted) vs. no such approach (black/solid).***

Torque Smoothness

Controller	$E[\|\Delta\tau\|]$	$E[\|\tau\|]$
ID$(t_i)_{QP,\mu}$	**52.5718**	**0.2016**
ID$(t_i)_{QP,\infty}$	245.5117	0.4239
ID$(t_i)_{LCP,\mu}$	677.9693	0.8351
ID$(t_i)_{LCP,\infty}$	351.1803	0.4055
ID(t_{i-1})	974.3312	0.9918
ID(t_{i-2})	17528.0	2.7416
PID	100.1571	0.3413

Table 13: *Average derivative torque magnitude (denoted $E[|\Delta\tau|]$) and average torque magnitude (denoted $E[|\tau|]$) for all controllers.*

7.7.1 Smoothness of torque commands Figure 57 shows the effects of an indeterminate contact model on torque smoothness. Derivative of torque commands are substantially smaller when incorporating a method of mitigating chatter in the inverse dynamics-derived joint torques. A five order of magnitude reduction was observed in the maximum of the time derivative torque when using a torque smoothing stage with the Drumwright-Shell contact model. Controller **ID$(t_i)_{LCP,\mu}$** is the only presented controller unable to mitigate torque chatter (seen in the "indeterminate" case in Figure 57) and therefore produces the worst performance from the presented inverse dynamics methods. Though it demonstrates successful behavior in simulation, this method would likely not

be suitable for use on a physical platform. The reference inverse dynamics controllers (**ID**(t_{i-1}) and **ID**(t_{i-2})) exhibit significant torque chatter also.

The "smoothness" of torque commands is measured as the mean magnitude of derivative torque over time. The data presented in Table 13 indicates that the two phase QP-based inverse dynamics controller (**ID**$(t_i)_{QP,\mu}$) followed by the baseline controller (**PID**) are the most suitable for use on a physical platform. Controller **ID**$(t_i)_{QP,\mu}$ uses the lowest torque to locomote while also mitigating sudden changes in torque that may damage robotic hardware.

7.7.2 Verification of correctness of inverse dynamics The correctness of each inverse dynamics approach is verified by comparing the contact predictions made by the controller against the reaction forces generated by the simulation. The comparison considers only the ℓ_1-norm of normal forces, though frictional forces are coupled to the normal forces (so ignoring the frictional forces is not likely to skew the results). Each experimental controller's contact force prediction is evaluated for accuracy given sticky and slippery frictional properties on rigid and compliant surfaces.

The QP-based controllers are able to predict the contact normal force in simulation to a relative error between 12–30%.[7] The **ID**$(t_i)_{LCP,\mu}$, **ID**$(t_i)_{LCP,\infty}$ controllers demonstrated contact force prediction between 1.16-1.94% relative error while predicting normal forces on a rigid surface (see Table 14). The QP based controllers performed as well on a compliant surface as they did on the rigid surfaces, while the performance of the **ID**$(t_i)_{LCP,\mu}$, **ID**$(t_i)_{LCP,\mu}$ controllers was substantially degraded on compliant surfaces.

The LCP-based inverse dynamics models (**ID**$(t_i)_{LCP,\mu}$ and **ID**$(t_i)_{LCP,\infty}$) use a contact model that matches that used by the simulator. Nevertheless no inverse dynamics predictions always match the measurements provided by the simulator. Investigation determined that the slight differences are due to (1) occasional inconsistency in the desired accelerations (the check described in Section 7.3.5 is not used); (2) the approximation of the friction cone by a friction pyramid in the experiments (the axes of the pyramid do not necessarily align between the simulation and the in-

[7] The QP-based inverse dynamics models use a contact model that differs from the model used within the simulator. When the simulation uses the identical QP-based contact model, prediction exhibits approximately 1% relative error.

verse dynamics model); and (3) the regularization occasionally necessary to solve the LCP (inverse dynamics might require regularization while the simulation might not, or *vice versa*).

Contact Force Prediction Error

Rigid, Low Friction		
Controller	absolute error	relative error
$\mathbf{ID}(t_i)_{QP,\mu}$	3.8009 N	12.53%
$\mathbf{ID}(t_i)_{QP,\infty}$	8.4567 N	22.26%
$\mathbf{ID}(t_i)_{LCP,\mu}$	0.9371 N	1.94%
$\mathbf{ID}(t_i)_{LCP,\infty}$	-	-
Compliant, Low Friction		
Controller	absolute error	relative error
$\mathbf{ID}(t_i)_{QP,\mu}$	7.8260 N	17.12%
$\mathbf{ID}(t_i)_{QP,\infty}$	-	-
$\mathbf{ID}(t_i)_{LCP,\mu}$	4.6385 N	6.37%
$\mathbf{ID}(t_i)_{LCP,\infty}$	-	-

Rigid, High Friction		
Controller	absolute error	relative error
$\mathbf{ID}(t_i)_{QP,\mu}$	13.8457 N	27.48%
$\mathbf{ID}(t_i)_{QP,\infty}$	12.4153 N	25.26%
$\mathbf{ID}(t_i)_{LCP,\mu}$	1.2768 N	1.55%
$\mathbf{ID}(t_i)_{LCP,\infty}$	0.3572 N	1.16 %
Compliant, High Friction		
Controller	absolute error	relative error
$\mathbf{ID}(t_i)_{QP,\mu}$	14.9225 N	30.86%
$\mathbf{ID}(t_i)_{QP,\infty}$	15.1897 N	30.66%
$\mathbf{ID}(t_i)_{LCP,\mu}$	12.4896 N	24.00%
$\mathbf{ID}(t_i)_{LCP,\infty}$	-	-

Table 14: *Average contact force prediction error (summed normal forces) of inverse dynamics controllers* vs. *measured reaction forces from simulation. The quadruped exerts 47.0882 N of force against the ground when at rest under standard gravity. Results marked with a "-" indicate that the quadruped was unable to complete the locomotion task before falling.*

7.7.3 Controller behavior The presented data supports the utilization of the QP-based inverse dynamics model incorporating Coulomb friction, at least for purposes of control of existing physical hardware. Utilizing Coulomb friction in inverse dynamics was observed to lead to much more stable locomotion control on various friction surfaces. The no-slip contact models proved to be more prone to predicting excessive tangential forces and destabilizing the quadruped while not offering much additional performance for trajectory tracking. Accordingly, subsequent results for locomotion on a height map and controlling a fixed-base manipulator while grasping a box are reported only for $\mathbf{ID}(t_i)_{QP,\mu}$ which can be referred to more generally as "the inverse dynamics controller with contact force prediction" or $\mathbf{ID}(t_i)$.

Rigid non-planar surface: Figure 55 plots trajectory tracking performance of the locomoting quadruped on rigid terrain with variable friction (ranging between low and high values of Coulomb friction for contacting materials as reported in literature). Three reference controllers are compared against the two-stage inverse dynamics controller. During this experiment only the ideal sensor controller $\mathbf{ID}(t_{i-1})$ consistently produced better positional tracking than the proposed controller

(**ID**(t_i)). The experimental controller reduced tracking error below that of error-feedback control alone by 19%.

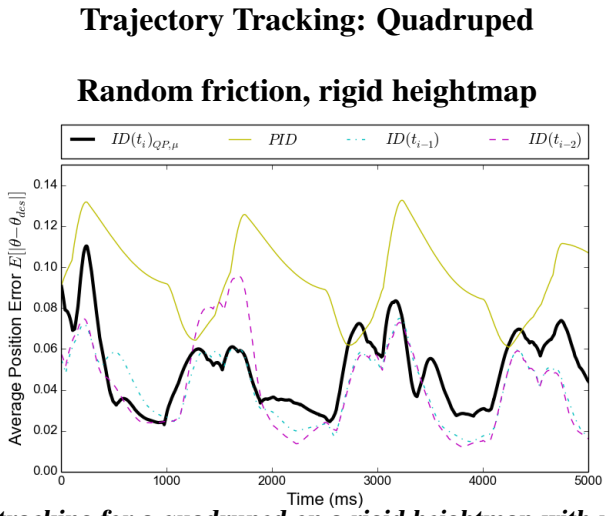


Figure 58: ***Joint trajectory tracking for a quadruped on a rigid heightmap with uniform random friction*** $\mu \sim \mathcal{U}(0.1, 1.5)$**.**

.

7.7.4 Center-of-mass tracking performance The ability of the controller to track the quadruped's center-of-mass over a path is a meta metric, as one expects this metric to be dependent upon joint tracking accuracy. Figure 59 shows that both the PID and the **ID**(t_i) methods are able to track straight line paths fairly well. **ID**(t_{i-1}), which yielded better joint position tracking, does not track the center-of-mass as well. **ID**(t_{i-2}) results in worse tracking with respect to both joint position and center-of-mass position. This discrepancy might be attributable to an earlier observation that **ID**(t_{i-1}) and **ID**(t_{i-2}) yield significantly larger joint velocity tracking errors than the PID and **ID**(t_i) controllers.

Fixed base manipulator grasping a box Trajectory tracking results for the fixed-base manipulator are presented in Figure 60. Large errors in the **PID** and **ID**(t_{i-1}) controllers were observed while grasping the box, the sensed force inverse dynamics controller was adversely affected when attempting to manipulate the sticky object, applying excessive forces while manipulating the box with high friction ($\mu = \infty$). Both the **PID** and **ID**$(t_i)_{QP,\mu}$ controllers dropped the box with low friction ($\mu = 1.0$) at about 1500 milliseconds. The trajectory error quickly converged to zero after the inverse dynamics method dropped the grasped object, while the **PID** controller maintained a

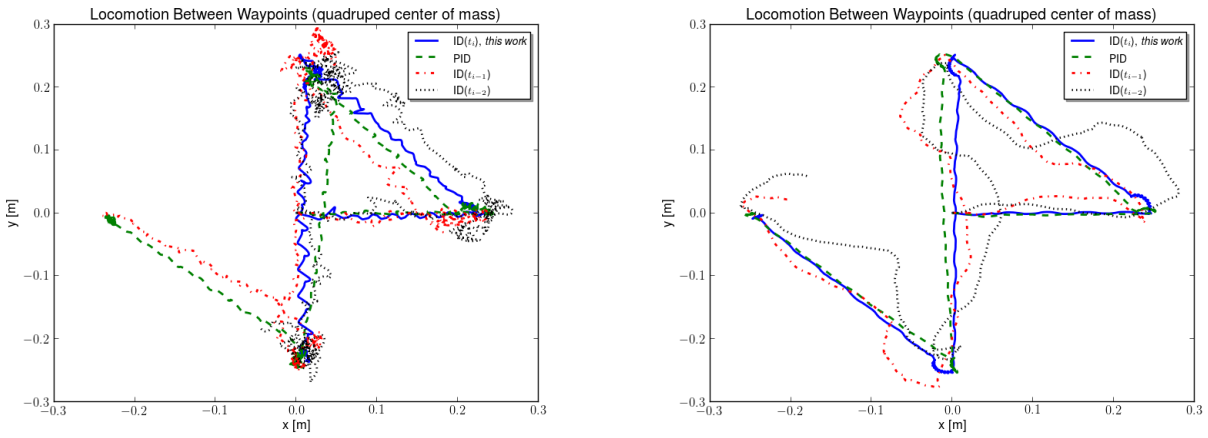
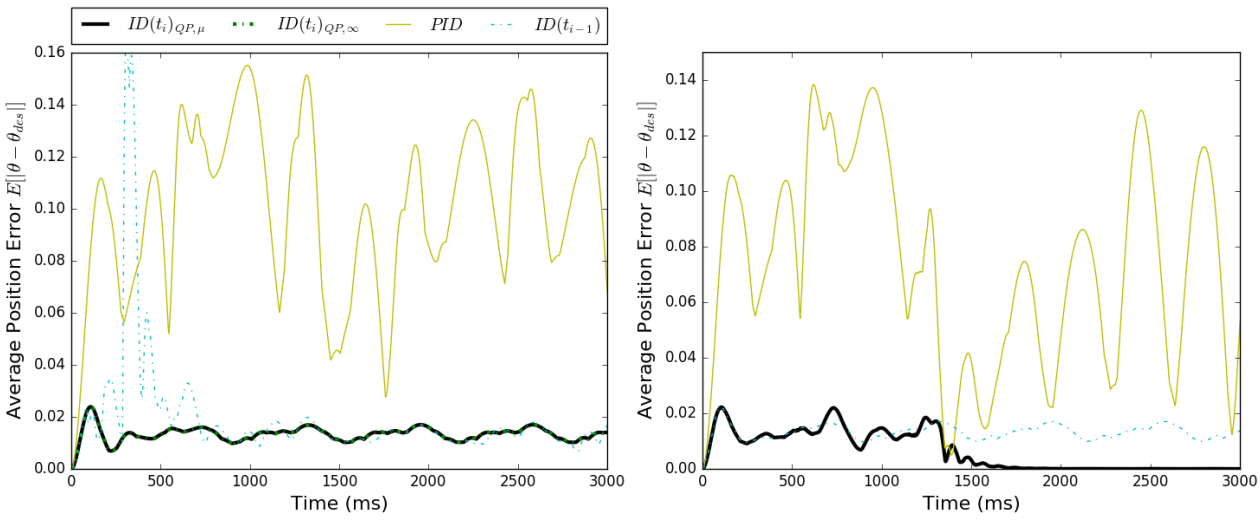


Figure 59: ***Center-of-mass path in the horizontal plane between waypoints over 30 seconds. (left) high friction; (right) low friction. The quadruped is commanded to follow straight line paths between points*** $\{(0,0),(0.25,0),(0,0.25),(0,-0.25),(-0.25,0)\}$**.**

relatively high level of positional error. The sensed contact inverse dynamics controller **ID**(t_{i-1}) performed at the same accuracy as the predictive contact force inverse dynamics controller, and managed to not drop the box over the course of the three second experiment.

Though the box slipped from the grasp of the inverse dynamics controlled manipulator, its tracking error did not increase substantially. This demonstrates a capability of the controller to direct the robot through the task with intermittent contact transitions with heavy objects, while maintaining accuracy in performing its trajectory-following task.



(a) ***High friction*** **($\mu = \infty$)*****, rigid surface***

(b) ***Low friction*** **($\mu = 1.0$)*****, rigid surface***

Figure 60: ***Joint trajectory tracking for a fixed base manipulator grasping a heavy box (6000*** $\frac{kg}{m^3}$***) with friction: (top)*** $\mu = \infty$***— no-slip; and (bottom)*** $\mu = 1$**.**

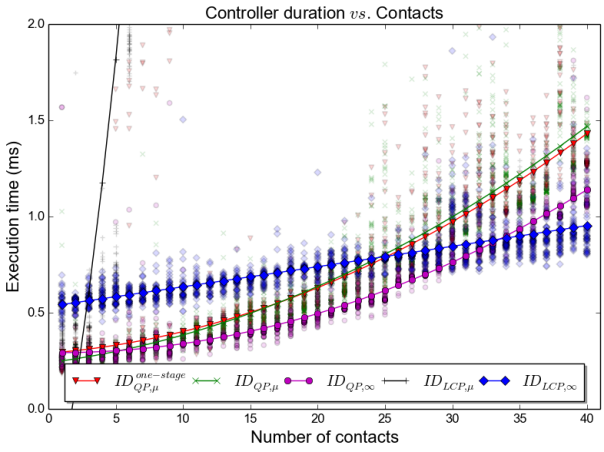


Figure 61: ***Inverse dynamics controller runtimes for increasing numbers of contacts (Quadruped with spherical feet).***

Running time experiments The computation time for each controller was measured during the quadruped experiments; the number of simultaneous contacts at each foot-ground interface was artificially increased to observe how each controller performed with a greater quantity to contacts to process.[8] Figure 61 shows that inverse dynamics method $\mathbf{ID}(t_i)_{LCP,\infty}$ scales linearly with additional contacts. The fast pivoting algorithm ($\mathbf{ID}(t_i)_{LCP,\infty}$) can process in excess of 40 contacts

[8]Experiments were performed on a 2011 MACBOOK PRO Laptop with a 2.7 GHz Intel Core i7 CPU.

while maintaining below a 1 ms runtime—capable of 1000 Hz control rate. The experimental QP-based controllers: $\mathbf{ID}(t_i)_{QP,\infty}$, $\mathbf{ID}(t_i)^{\text{one-stage}}_{QP,\mu}$, $\mathbf{ID}(t_i)_{QP,\mu}$ supported a control rate of 1000 Hz to about 30 total contacts when warm-starting the LCP solver with the previous solution. Controller $\mathbf{ID}(t_i)_{LCP,\mu}$ did not support warm-starting or the fast pivoting algorithm and was only able to maintain around a 1 ms expected runtime for fewer than 4 contacts. The runtime for $\mathbf{ID}(t_i)_{LCP,\infty}$ was substantially higher than the QP-based model, despite a significantly reduced problem size, for fewer than approximately 30 points of contact. This disparity is due to the high computational cost of Lines 6 and 10 in Algorithm 3.

7.7.5 Discussion of inverse dynamics based control for legged locomotion The inverse dynamics controller that predicts contact forces ($\mathbf{ID}(t_i)_{QP,\mu}$) performs well, at least in simulation, while mitigating destructive torque chatter. Over all tests, the use of inverse dynamics control with predicted contact forces, *i.e.*, $\mathbf{ID}(t_i)$, was observed to more closely track a joint trajectory than the alternatives—including inverse dynamics methods using sensed contact forces (even with perfect sensing) and PID control. The inverse dynamics controller described in this work $\mathbf{ID}(t_i)$ and the baseline PID controller, were able to track center-of-mass position of quadrupeds with little deviation from the desired direction of motion, while the inverse dynamics controllers using sensed contact forces performed substantially worse at this task (as seen in Section 7.7.4).

This chapter presented multiple, fast inverse dynamics methods—a method that assumes no slip (extremely fast), a QP-based method without complementarity (very fast), that same method with torque chattering mitigation (fast enough for real-time control loops at 1000Hz using current computational hardware on typical quadrupedal robots), and an LCP-based method that enforces complementarity (fast). It was demonstrated that a right inverse exists for the rigid contact model with complementarity, and the asymptotic time complexity was analyzed for all inverse dynamics methods. Each method is likely well suited to a particular application. For example, Section 7.7 found that the last of these methods yields accurate contact force predictions (and therefore better joint trajectory tracking) for virtual robots simulated with the Stewart-Trinkle/Anitescu-Potra contact models. Finally, the performance of each controller was assessed—running times, joint-space

trajectory tracking accuracy, and an indication of task execution capability (for legged locomotion and manipulation)—under various contact modeling assumptions.

7.8 Conclusion

Inverse dynamics can be a highly effective method in multiple contexts including control of robots and physically simulated characters and estimating muscle forces in biomechanics, but requires knowledge of all external forces acting on the robot. Accurately perceiving external forces applied to a robot (to enable accurate inverse dynamics control) requires filtering and thus significant time delay. An alternative approach of predicting contact and actuator forces simultaneously under the assumptions of rigid body dynamics, rigid contact, and friction was presented in this chapter. The presented controllers (rated on performance in Appendix E) can form the basis of an effective nonlinear control strategy in manipulator and legged robots by providing a robot with both accurate positional tracking and active compliance. In biomechanics applications, inverse dynamics control can approximately determine the net torques applied at anatomical joints that correspond to an observed motion.

Results derived from simulation used ideal sensing and perfect torque control. The controllers presented in this chapter permit the virtual robotic tests in Chapters 4 and 6 to exhibit the limit of performance for each assessed robotic system.

8 Numerical stability for simulating robots controlled with error feedback

Roboticists often wish to simulate controlled systems rapidly while prototyping control schemes and hardware designs. With sufficiently fast simulation, the robot virtual testing approach from Chapter 4 could be executed *online* as in Section 4.3.3 and the robot prototyping process from Chapter 6 could be completed in seconds rather than minutes. For example, when designing a new gait generation and balance stabilization software for a simulated legged robot, the underactuated nature of the robot limits the utility of strictly kinematic simulation (e.g., a robot will not exhibit a fall when testing a bad balancing controller if not dynamically simulated). Dynamic simulation of a virtual robot requires the inclusion of error feedback controllers, which then necessitate careful tuning to balance dynamic performance with simulation stability. These additional steps make debugging both more challenging (e.g., Did the robot fall because control was not sufficiently accurate?) and slower than desirable—with the inclusion of control, inverse kinematics, and planning code, simulations would run several times slower than real-time. In these cases, reducing the duration of the "edit-compile-test" cycle might be more important than reducing numerical solution error, which roboticists might do after obtaining some confidence in their control, planning, and estimation software.

The work in this chapter flowed from a previous investigation (Zapolsky & Drumwright, 2015) into a means of accelerating this process. That research found that exponential energy dissipation can be used to increase simulation stability. However, it should be clear that excessive dissipation will lead to artifacts (e.g., the robot acts as if it is moving through molasses); it is challenging to balance numerical stability and physical fidelity with that approach. Experiments from Chapter 7 indicated that inverse dynamics controllers admitted larger integration steps than robots controlled with error feedback control, which exhibited "stiff" behavior (see §8.1). This chapter presents two alternative approaches to the work in Zapolsky & Drumwright 2015 that admit large integration steps for various simulation robotic system; the presented approaches incorporate the following into the simulator's equations of motion for a robotic system: (1) prescribed motion constraints

such as a desired acceleration in an inverse dynamics control formulation (§8.4); and (2) transmission models for electromagnetic actuators (§8.5.3). Accordingly, this chapter tests the following hypotheses:

Hypothesis A: Driving a multi-body system (e.g., a robot interacting through contact with one or more rigid bodies) through inverse dynamics control can yield greater numerical stability than tuned PD/PID control can offer.

Hypothesis B: Integrating the equations of motion for multi-body systems that accounts for both contact and prescribed motion constraints yields greater numerical stability than simply feeding the output from an inverse dynamics controller into the simulation's integrator.

Hypothesis C: Incorporating transmission models for electromagnetic actuators into the equations of motion can increase numerical stability for simulations of robotic systems driven by PD/PID control.

8.1 "Stiff" systems

An informal characterization of "stiff" dynamical systems is that their solution is smooth, yet explicit numerical approaches to solving initial value problems with them can require extremely small integration steps to capture the desired behavior. Mechanical systems with springs and dampers are the archetype of stiff systems, and error feedback controllers applied to mechanical systems without springs and dampers act stiff as well. As a result, rapid prototyping of systems driven with roboticists' typical control techniques has proven challenging; simulations may run one or more orders of magnitude more slowly than useful in such preliminary testing.

8.1.1 High mass ratios Multi-rigid body systems with high mass ratios between links can be viewed as a type of stiff system (Anitescu & Potra, 2002). However, Featherstone found that the condition number of the joint space inertia matrix increases quartically with the length of a kinematic chain (Featherstone, 2004), which points to another possible source of system stiffness. The robotic systems used in the experiments within this present work were modeled from CAD data and do not contain significant disparities in masses, yet the inertia matrix condition numbers

(on the order of 10^8) are still problematic.

8.2 "Motors" in Open Dynamics Engine

The OPEN DYNAMICS ENGINE (ODE) manual describes "motors", which implement the technique described in this present work, in the following way.

> ... [A]pplying forces directly [to joints] is often not a good approach and can lead to severe stability problems if it is not done carefully.
>
> Consider the case of applying a force to a body to achieve a desired velocity. To calculate this force F you use information about the current velocity, something like this:
>
> $$F = k(\text{desired speed} - \text{current speed}) \tag{147}$$
>
> This has several problems. First, the parameter k must be tuned by hand. If it is too low the body will take a long time to come up to speed. If it is too high the simulation will become unstable. Second, even if k is chosen well the body will still take a few time steps to come up to speed. Third, if any other "external" forces are being applied to the body, the desired velocity may never even be reached (a more complicated force equation would be needed, which would have extra parameters and its own problems).
>
> Joint motors solve all these problems ... They can effectively see one time step into the future to work out the correct force. This makes joint motors more computationally expensive than computing the forces yourself, but they are much more robust and stable, and far less time consuming to design with. This is especially true with larger rigid body systems.

This text indicates that incorporating prescribed motions into the constraint equations is convenient for the user, but it does not elaborate upon the difficulty of attaining fast simulation performance even with well-tuned error feedback control. A general writeup of this technique is not

readily available to the robotics community, including its incorporation into the differential variational inequality formulation.

8.3 Kinematic simulations

The Institute for Human and Machine Cognition's (IMHC) robotics group has adopted a similar approach to that described in this chapter:[9] desired joint and floating base accelerations are double integrated to yield kinematically driven simulations of legged robots for rapid testing. This approach apparently works well for robots interacting with static environments, but has been tested little on robotic manipulation tasks. The differences between IHMC's approach and the inverse dynamics and prescribed motion approaches described in this chapter follow: (1) inverse dynamics and prescribed motion constraints retain floating base underactuation, allowing a legged robot to trip, for example (IHMC's approach would not capture this behavior); (2) Simulation with non-interpenetration and torque constraints, among other constraints, precludes dynamically infeasible motions (to the first-order accuracy provided by the time stepping-based approach); and (3) IHMC's approach does not have to constrain the motion by solving optimization or mathematical programming problems, meaning that it will operate far faster.

8.4 Multi-body dynamics simulation with contact and inverse dynamics

This section presents the formulation of the time stepping multi-body dynamics problem with contact and prescribed motion constraints. Joint range of motion limits and bilateral constraints can be incorporated using straightforward extensions and are not discussed here to streamline the presentation. This *mixed linear complementarity problem formulation* (Cottle et al., 1992) is used in ODE and other software.

[9] Personal communication with Jerry Pratt.

$$\begin{bmatrix} \mathbf{M} & -\mathbf{P}^{\mathsf{T}} & -\mathbf{N}^{\mathsf{T}} & -\mathbf{F}^{\mathsf{T}} & \mathbf{0} \\ \mathbf{P} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{F} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E} \\ \mathbf{0} & \mathbf{0} & \mu & -\mathbf{E}^{\mathsf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^{+} \\ \boldsymbol{\tau} \\ \boldsymbol{f}_N \\ \boldsymbol{f}_F \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{\kappa} \\ -\dot{\boldsymbol{q}}_{\text{des}} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{w}_{\boldsymbol{\tau}}^{+} - \boldsymbol{w}_{\boldsymbol{\tau}}^{-} \\ \boldsymbol{w}_{\boldsymbol{N}} \\ \boldsymbol{w}_{\boldsymbol{F}} \\ \boldsymbol{w}_{\boldsymbol{\lambda}} \end{bmatrix} \tag{148}$$

$$\boldsymbol{\tau} - \boldsymbol{\tau}^{-} \geq \mathbf{0}, \boldsymbol{w}_{\boldsymbol{\tau}^{+}} \geq \mathbf{0}, \left(\boldsymbol{\tau} - \boldsymbol{\tau}^{-}\right)^{\mathsf{T}} \boldsymbol{w}_{\boldsymbol{\tau}^{+}} = 0 \tag{149}$$

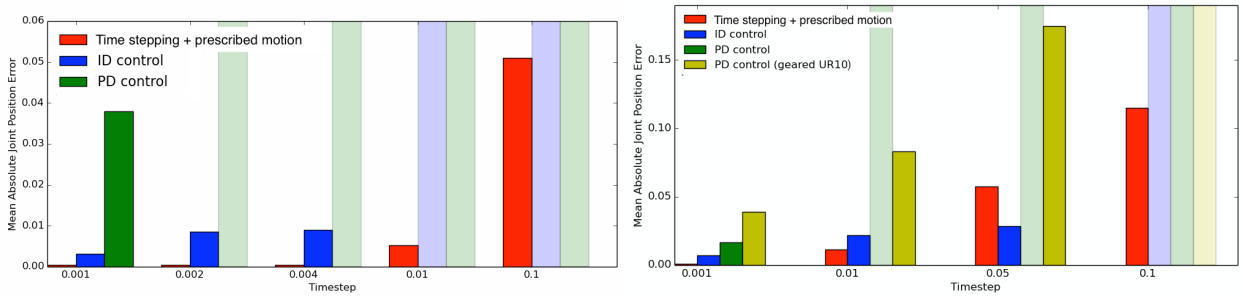
$$\boldsymbol{\tau}^{+} - \boldsymbol{\tau} \geq \mathbf{0}, \boldsymbol{w}_{\boldsymbol{\tau}^{-}} \geq \mathbf{0}, \left(\boldsymbol{\tau}^{+} - \boldsymbol{\tau}\right)^{\mathsf{T}} \boldsymbol{w}_{\boldsymbol{\tau}^{-}} = 0 \tag{150}$$

$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{w}_N \geq \mathbf{0}, {\boldsymbol{f}_N}^{\mathsf{T}} \boldsymbol{w}_N = 0 \tag{151}$$

$$\boldsymbol{f}_F \geq \mathbf{0}, \boldsymbol{w}_F \geq \mathbf{0}, {\boldsymbol{f}_F}^{\mathsf{T}} \boldsymbol{w}_F = 0 \tag{152}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{w}_\lambda \geq \mathbf{0}, \lambda^{\mathsf{T}} \boldsymbol{w}_\lambda = 0 \tag{153}$$

This problem formulation matches that in Chapter 7 almost exactly, and the reader is referred to that work for greater insight into inverse dynamics subject to unilateral constraints. Briefly, there are m generalized velocities (r of which are actuated), n points of contact, and k line segments in each polygonal approximation to the friction cone at each point of contact; problem inputs are $\mathbf{M} \in \mathbb{R}^{m \times m}$ (generalized inertia matrix), $\boldsymbol{v} \in \mathbb{R}^m$ (generalized velocity vector), $\mathbf{P} \in \mathbb{R}^{r \times m}$ is a binary selection matrix (the identity matrix if the controlled system is fully actuated), $\mathbf{N} \in \mathbb{R}^{n \times m}$ (contact normals Jacobian matrix), $\mathbf{F} \in \mathbb{R}^{nk \times m}$ (contact tangents Jacobian matrix), $\mu \in \mathbb{R}^{n \times n}$ (diagonal matrix of friction coefficients), $\mathbf{E} \in \mathbb{R}^{nk \times n}$ (binary matrix described in Anitescu & Potra 1997), $\dot{\boldsymbol{q}}_{\text{des}} \in \mathbb{R}^r$ (desired actual joint velocities), and $\boldsymbol{\kappa} \equiv \mathbf{M}\boldsymbol{v} + \Delta t \boldsymbol{f}$ ($\boldsymbol{f} \in \mathbb{R}^m$ are all non-contact and non-inverse dynamics forces on the system, Δt).



(a) ***Quadruped*** **(b)** ***UR10***

Figure 62: ***The mean absolute joint position error for the ID constraint, ID control, and PD control for the quadruped and UR10 robots at various timesteps. A transparent bar indicates instability for the control at the given timestep.***

Key solution variables are $\boldsymbol{v}^+ \in \mathbb{R}^m$ (velocities at $t + \Delta t$, i.e., after integration), $\boldsymbol{\tau} \in \mathbb{R}^r$ (inverse dynamics forces/torques), $\boldsymbol{f}_N \in \mathbb{R}^n$ (contact normal forces), $\boldsymbol{f}_F \in \mathbb{R}^{nk}$ (contact friction forces), $\boldsymbol{\lambda} \in \mathbb{R}^n$ (roughly equivalent to tangent contact velocities at $t + \Delta t$, see Anitescu & Potra 1997). Variables $\boldsymbol{v}^+$ and $\boldsymbol{\lambda}$ are unbounded, maximum actuator forces $\boldsymbol{\tau}$ are constrained within the interval $[\boldsymbol{\tau}^-, \boldsymbol{\tau}^+]$, and normal $\boldsymbol{f}_N$ and frictional $\boldsymbol{f}_F$ contact force variables are constrained within the half-closed interval of non-negative values $[0, \infty)$.

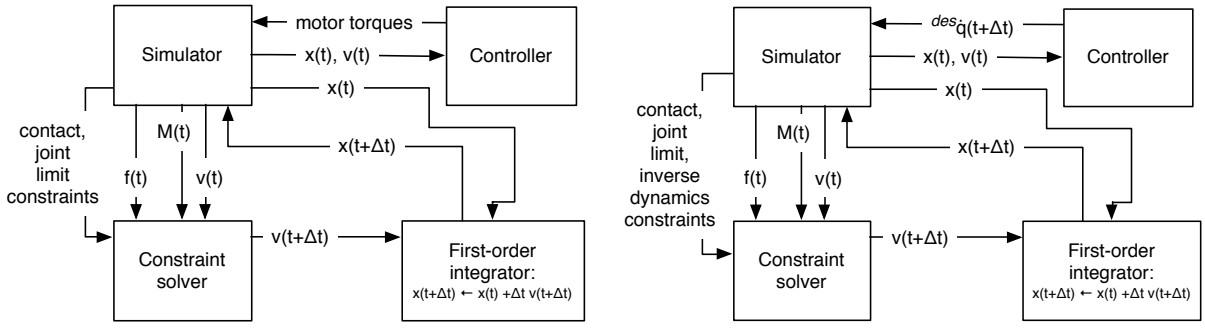
8.4.1 Drawback of the MLCP formulation The difference between the inverse dynamics control algorithms presented in Chapter 7 and ODE's approach is that $\boldsymbol{\tau}$ is constrained to lie in $[\boldsymbol{\tau}^-, \boldsymbol{\tau}^+]$ in the latter, which means that the prescribed motion constraints might not be perfectly satisfied (hence the presence of the term $\boldsymbol{w}_{\boldsymbol{\tau}}^+ - \boldsymbol{w}_{\boldsymbol{\tau}}^-$). These new equations and variables specify that if the prescribed motion constraint can be satisfied exactly, then $\boldsymbol{w}_{\boldsymbol{\tau}}^+ - \boldsymbol{w}_{\boldsymbol{\tau}}^- = \mathbf{0}$. Otherwise, the joint torque acts against the "slack" in the constraint. In other words, if the velocity at $\boldsymbol{v}^+$ for the i^{th} joint is greater than that desired, then the torque applied at that joint must lie at the lower limit; similarly, the torque applied at that joint must lie at the upper limit if the velocity at $\boldsymbol{v}^-$ for the i^{th} joint is lesser than that desired. This problem setup is reasonable in many but not all cases: it is possible that applying a torque at an actuator's lower torque limit could result in greater divergence from the desired velocity at that joint than if no torques were applied (depending on other variable settings). Neither does this problem setup appear to minimize any norm over the difference between desired

and resulting velocity.

8.4.2 Solvability of the MLCP formulation Chapter 7 will show that this problem can be solved in expected polynomial time, including determining whether the desired velocities are consistent with the other constraints, for $\boldsymbol{\tau}^- \equiv -\infty, \boldsymbol{\tau}^+ \equiv \infty$ by first converting it to a "pure" linear complementarity problem. For finite torque limits, the mixed linear complementarity problem cannot be converted to a pure LCP, so an algorithm for solving mixed LCPs of this form must be applied. Lemke's Algorithm can be modified to handle lower and upper variable limits (as described in Sargent 1978) but is only provably able to solve MLCPs with positive semi-definite matrices. *The result is that the MLCP in Equations 148–153, which results in a copositive matrix (Stewart & Trinkle, 2000; Cottle et al., 1992), is not generally capable of being solved in polynomial time using existing algorithms.* In fact, when the desired velocities are inconsistent with the other constraints, the MLCP is unsolvable even without torque limits. Accordingly, ODE uses regularization to solve a "nearby" MLCP. All problem constraints—including non-interpenetration, Coulomb friction, joint limits, and inverse dynamics—will be violated by the degree of regularization. The ramifications of such violations are generally unknown, though an experiment in §8.5.2 describes one outcome.

8.5 Experiments

Simulation experiments were conducted using the multi-rigid body dynamics library MOBY, which uses pivoting solvers in place of the matrix splitting method solvers often employed to speed simulations; these solvers do not provably converge (Lacoursière, 2003), so they are avoided to simplify these experiments. The virtual robots used in the experiments were the UR10 arm (sourced from an existing open source ROS package) and a floating base quadruped model described Chapter 4. The UR10 arm possesses eight degrees-of-freedom (DoF), all controllable. The quadruped model possesses 18 DoF, 12 of which are controllable. The UR10 was directed to follow a sinusoidal motion at each joint, while the quadruped was commanded to follow a sinusoidal pattern that resembled a trot. Each simulation was run for five seconds of virtual time.



(a) ***Inverse dynamics forces fed into the simulator***

(b) ***Simulator computing the next velocity of the multi-rigid body dynamics simulation while accounting for the prescribed motion constraints***

Figure 63: ***The difference between different prescribed motion approaches in simulation. x, v, M, and f are the generalized positions, velocities, inertias, and forces respectively. The architecture on the right is faster, because that on the left (invrse dynamics) solves essentially the same problem twice: once in the controller—the contact forces must be accounted for to computer the inverse dynamics forces, but the former are then discarded—and then again in the simulator.***

When PD controllers were used in the following experiments, gains were tuned by a two-part process consisting of manual tuning followed by nonlinear optimization. The optimization routine minimized the ℓ_2-norm over the sum of all squared joint position errors. The gains were tuned in this way to eliminate human bias to the greatest extent possible; otherwise, the experimenter could subconsciously reduce gains to prioritize simulation stability over tracking accuracy.

Where applicable, the experiments in this section used algorithms for computing prescribed motion under constraints that *can* account for torque limits—the ones described in this chapter, not in Chapter 7—even though torque limits are set to $\pm\infty$ (i.e., unused). This decision permitted the study of the computational properties of these methods without focusing on whether the virtual robots would possess the actuator forces necessary to execute the designated tasks. Integration steps were limited to a maximum of 0.1s. $(O)(h)$ terms in any integrator imply very large errors for $h \geq 1$. The initial integration step size tested for each model was 0.001, a value that typically produced stable simulations of each model. Step sizes were doubled until instability resulted—at which point bisection search was conducted to identify an approximate maximum stable step size—or the maximum value of 0.1s was reached.

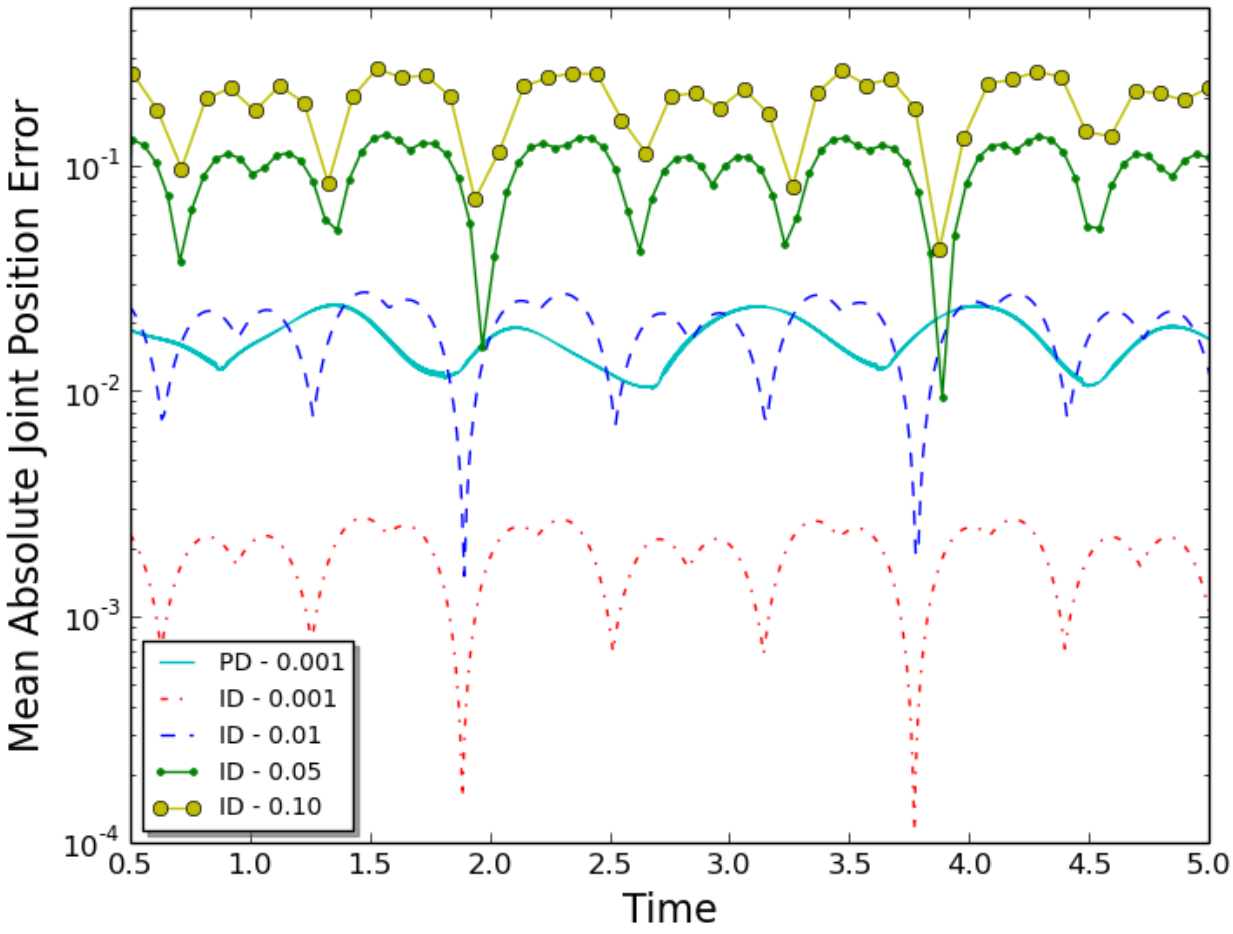


Figure 64: ***The mean absolute position error of each joint on the UR10 arm as it executes a sinusoidal motion on each joint at various timesteps. Time stepping with prescribed motion was used to achieve the maximum 0.1 timestep.***

8.5.1 Testing the hypothesis that incorporating inverse dynamics control leads to more stable simulations than error feedback control

A PD-controlled UR10 arm served as one experimental control for Hypothesis A. The maximum step size attained using this controller without the simulation becoming unstable was 0.001. On the other hand, it was found that the UR10 arm could be simulated stably without any controls applied (i.e., the robot falls under the influence of gravity) with a step size of 0.1 (the maximum tested). *This result indicates that the PD control introduces stiffness into the differential equations*; this result is not surprising given that (1) PD control can be viewed as a virtual spring damper and (2) the mass-spring system is the canonical example of a stiff ordinary differential equation (Hairer & Wanner, 1996). For the experimental variable,

the standard recursive Newton-Euler Algorithm (Featherstone, 1987) was used to generate inverse dynamics torques.

Control method	Max step	Accuracy at max step	Running time
PD control	0.001	3.80×10^{-2}	24.84s
Inverse dynamics (control)	0.004	9.05×10^{-3}	4.97s
Time stepping with prescribed motion (MLCP approach)	—	—	—
Time stepping with prescribed motion (experimental solver)	0.10	5.10×10^{-2}	2.54s

Table 15: *Mean of absolute error joint position tracking* accuracy *on the simulated quadrupedal robot*

An experimental control in a second experiment used the quadrupedal robot model driven by PD control. The non-complementarity-based inverse dynamics control approach described in Chapter 7 was used as the experimental variable, which was necessary since one or more links of the robot remained in contact with the environment. Tables 15 and 16 list the results from the experiments with both robots: time stepping with prescribed motion allows the simulations to run 540% and 66% faster, respectively.

In both experiments, the PD control yielded the smallest stable integration time steps. Figure 62a shows that ID control was able to achieve a maximum stable timestep of 0.004 during the quadruped experiment. PD control, on the other hand was only able to achieve a maximum timestep of 0.001. Figure 62b shows the ID control being able to achieve a timestep of 0.05 when the sinusoidal motion was run on the UR10 arm. PD control was still only able to achieve a maximum step size of 0.001. The large increase of timestep with regards to the ID control from the quadruped experiment to the UR10 experiment is likely due to the fact that the quadruped actually experiences contact in its controlled motion, while the UR10 experiences no contact.

8.5.2 Testing the hypothesis that incorporating prescribed motion constraints leads to more stable simulations than using inverse dynamics control Although it is possible that inverse dynamics torques fed into a simulator yield exactly the desired velocity at the next time step[10], this

[10]desired velocity was used in place of desired acceleration for reasons described in Chapter 7. Popular open source multi-rigid body dynamics libraries used for robotics (e.g., ODE, BULLET, DART) employ a first-order approxima-

result is not guaranteed (as the data from the previous section show). Hypothesis B arose from observations about the split nature of the control-simulation process (see Figure 63): it was speculated that solving for the next velocity subject to all constraints would yield higher greater simulation stability than feeding the inverse dynamics torques into the simulator's constraint solver (i.e., its mixed or pure linear complementarity problem solver).

Inverse dynamics controllers were employed as the experimental variables in the tests of Hypothesis A as the experimental controls in the tests of Hypothesis B. For the variables in this experiment, UR10 arm was tested and the quadrupedal robot using the MLCP-based inverse dynamics formulation described in Section 8.4. The quadrupedal robot was also tested using an experimental, optimization-based constraint solver. Only an outline of the technical approach will be outlined as a key problem with the approach was identified during experimentation (to be described below). The solver first computes a feasible point that satisfies both the contact normal velocity constraints ($\mathbf{N}\boldsymbol{v}^+ \geq \mathbf{0}$, from Equation 148) and the inverse dynamics velocity constraints ($\boldsymbol{P}\boldsymbol{v}^+ - \dot{\boldsymbol{q}}_{\text{des}} = \mathbf{0}$). Quadratic programming is then used to attempt to find frictional forces that maximally dissipate kinetic energy without violating these constraints (in the spirit of Drumwright & Shell 2010).

Figure 64 depicts the speedup achieved from the use of time stepping with prescribed motion constraints. At a timestep of 0.01, the use of prescribed motion constraints is able to achieve around the same order of tracking accuracy as the PD control at a timestep of 0.001. Incorporating prescribed motion constraints into the UR10 simulation process executed in 9.11s at a 0.01 timestep, while the PD controlled arm required 357.02s at a 0.001 timestep. Time stepping with prescribed motion constraints achieved a 39x speedup with essentially the same accuracy.

The results for the quadrupedal robot in Table 15 require explanation. The existing, MLCP-based approach caused the simulation to become unstable at any step size. Regularizing the MLCP (via Tikhonov regularization) did not help: such large regularization—it was necessary to add values on the order of 1.0 to MLCP matrix to attain a solution while typical values lie in $[10^{-12}, 10^{-8}]$—that the result was no longer a solution to a "nearby" problem. Note that applying the constraint solver

tion to velocity, thus supporting the choice.

to the individual problems of prescribed motion constraints without contact (by temporarily deactivating gravitational forces) and contact without prescribed motion constraints (i.e., just using PD control) works fine; problems only arise when the constraints are considered simultaneously. The Dantzig MLCP solver (Lacoursière, 2007) was used, which is also used by ODE, to solve the mixed linear complementarity problem.

On the other hand, the experimental approach outlined above was capable of generating an accurate solution—and, as with the UR10 model—the simulation remained perfectly stable for large step sizes. However, the presented results do not capture an important artifact: the quadruped appeared to be skating as if on ice when it should have been trotting. Examination of the constraint solver indicated that the solution method would have had to slightly violate the prescribed motion constraints to incorporate frictional forces; the presented experimental approach is flawed and thus illustrates what can happen when constraints may be violated arbitrarily (refer back to §8.4.2). Nevertheless, Table 15 does hint at the possibility of a 226% speedup.

Control method	Max step	Accuracy at max step	Running time
PD control + simulation	0.001	1.01×10^{-2}	357s
PD control (geared robot) + simulation	0.05	1.74×10^{-1}	24.6s
Inverse dynamics (control) + simulation	0.05	2.85×10^{-2}	6.51s
Time stepping with prescribed motion	0.10	1.15×10^{-1}	2.85s

Table 16: *Mean of absolute error joint position tracking* accuracy *on the simulated UR10 manipulator*

8.5.3 Testing hypothesis that incorporating transmission models increases the stability of robots driven by error feedback control Claude Lacoursière suggested in personal communication that adding gearing to a robot model might reduce the computational stiffness in the differential equations. Accordingly, the PD controlled UR10 used to test Hypothesis A was compared to a PD controlled UR10 with a virtual transmission modeled at each revolute joint; the gains were re-tuned for this modified model. Gearing was not added to the quadrupedal model because significant architectural modifications would be necessary in the robot's locomotion software to accommodate

gearing. Table 16 and Figure 64 illustrate that the gearing does dramatically increase the maximum stable step size, at a clear cost of tracking accuracy.

8.5.4 Discussion of Results This chapter has demonstrated the capability of inverse dynamics and prescribed motion to dramatically speed multi-rigid body simulations with contact. Each prescribed motion constraint (i.e., specification of a joint velocity) increases the size of the mixed LCP (when accounting for force/torque limits) or pure LCP (without force/torque limits) to be solved; in the latter case, a variable is added to a linear system to be solved independently of the LCP, as described in Chapter 7.[11]

Aside from the wasted computational effort of solving the same problem twice—once for the controller to compute inverse dynamics subject to contact and joint limit constraints and once for the simulation's solver to compute contact and joint limit forces subject to the control forces/-torques, see Figure 63—this chapter has shown that incorporating the constraints into the constraint solver is less likely to cause simulation instability. The stability decrease from feeding the inverse dynamics forces/torques into the simulation might be explained by very slight discrepancies in inputs; for example, the friction pyramids used for the Coulomb friction approximations between the two constraint solvers can use different principal directions (which are selected arbitrarily). The previous chapter, Chapter 7 demonstrates high, albeit imperfect tracking accuracy for inverse dynamics of simulated robots, hinted at this phenomenon.

8.6 Conclusion

Although the computational demands to solve these problems are larger than that required for error feedback controlled robots, the maximum stable integration step sizes are tens or hundreds of times larger, thereby permitting much faster simulations. The complementarity-free contact model described in Drumwright & Shell 2010 might provide a foundation for a computationally tractable model that produces reasonably accurate contact forces and satisfies prescribed motion constraints as well as force/torque limits allow. In the meantime, adding gearing to robots with electromagnetic

[11] this is exactly the same procedure as that required to account for a gear constraint; the computational demands are identical.

actuators can provide the requisite simulation stability necessary for high frequency (realtime and above) simulation, albeit with far lower tracking accuracy.

Advancements toward simultaneously improving the accuracy and stability (and correspondingly simulator speed) such as those presented in this chapter might permit roboticists to simulate controlled systems rapidly while prototyping control schemes and hardware designs. Rapid simulation will reduce the duration of the "edit-compile-test" cycle of the robot prototyping process (Chapter 6) permitting roboticists to quickly edit and test a robot design in an interactive design environment and then verify the robustness of the design through virtual testing (Chapter 4).

9 Quadrupedal Robot Locomotion

The software described in this chapter, PACER, implements the software described in this thesis (e.g., inverse dynamics controllers, online particle traces) and basic functionality plugins for typical robot operation (e.g., inverse kinematics, configuration space error-feedback control, operational space error-feedback control). PACER was developed to control walking and running quadrupedal robots *in situ* and features a reactive gait planning plugin that is the primary focus of this chapter.

Similar compilations of robotics planning and control tools have been made publicly available, including the work of the Gepetto team METAPOD[12] from LAAS CNRS and DRAKE[13] from the Robot Locomotion Group at MIT (Tedrake & the Drake Development Team, 2016). PACER is a package developed for real-time perception, planning, and control of simulated and physical robots. PACER provides:

A high level control interface in Section 9.1 that abstracts planning and control for legged robot locomotion to high level commands (e.g., Dubin's car, Reed-Shepp car, etc.).

A reactive gait planner in Section 9.2 that .

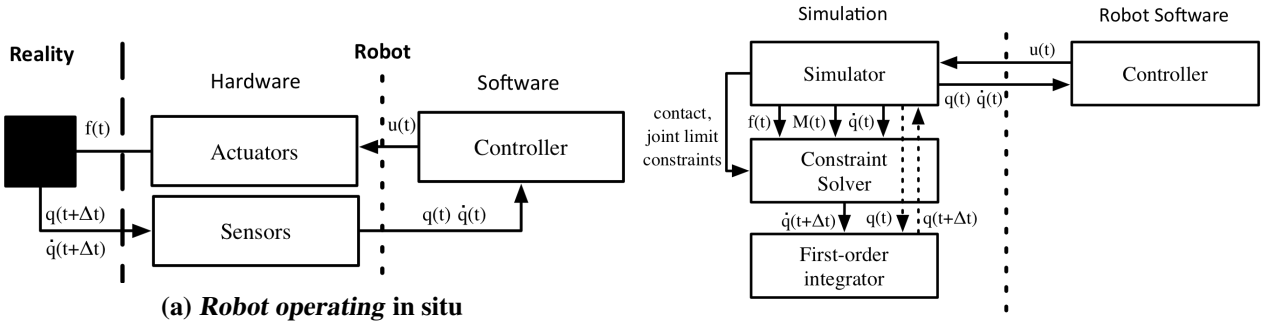
A plugin robot and controller architecture in Section 9.3 for seamless transition of control from *in situ* to simulated robots (also used by Schaal 2009). It permits hot-swapping controllers while a robot is active.

9.1 Locomotion control policy

The PACER library fills the role of "Robot Software" in Figure 65. More in-depth depictions of the relationship between robotic software and a robot (real or simulated) were previously illustrated in Figures 1 and 2; the "Controller" in these diagrams describes the most basic job of a control policy: (1) receive a robot's state $\{\boldsymbol{q}(t), \dot{\mathrm{q}}(t)\}$; and (2) output commands for the robot $\boldsymbol{u}(t)$ (e.g., actuator torques).

[12] available at https://github.com/laas/metapod

[13] available at https://github.com/RobotLocomotion/drake



(a) *Robot operating* in situ

(b) *Robot operating* in sim

Figure 65: ***A flow chart depicting how* PACER *(robot software) interacts with (a) a robot* in situ *or (b) a time-stepping simulator. The software receives a robot's state as input and outputs actuator torques.***

Input forces $\boldsymbol{u}(t)$ are determined by a control policy π, which is dependent on the current state of the system $\{\boldsymbol{q}(t), \dot{\mathrm{q}}(t)\}$, time t, and the interval until the next controller call Δt, followed by a list of policy-specific parameters (e.g., control policy parameters $\boldsymbol{p}$ and desired spatial velocity of the robot base $\dot{\mathrm{x}}_{\mathrm{des}}^{\mathrm{base}}$ in Equation).

$$\boldsymbol{u}(t) = \pi\left(\boldsymbol{q}(t), \dot{\mathrm{q}}(t), t, \Delta t, ...\right) \tag{154}$$

The time-step Δt is necessary to consider when incorporating inverse dynamics control into a control policy that is formulated in the continuous domain. The inverse dynamics controllers used in this dissertation are formulated with respect to a first order approximation of the dynamics of the robotic system; they must know over what interval of time to evaluate input forces to the robotic system to reach the desired velocity input to the controller (refer to Chapter 7).

$$\boldsymbol{u}(t) = \pi_{\mathrm{locomotion}}(\boldsymbol{q}(t), \dot{\mathrm{q}}(t), t, \Delta t, \boldsymbol{p}, \dot{\mathrm{x}}_{\mathrm{des}}^{\mathrm{base}}) \tag{155}$$

Algorithm 6 describes the evaluation of Equation 155 in PACER for a locomoting quadrupedal robot.

Algorithm 6 $\{\boldsymbol{u}(t)\} = \pi_{\text{locomotion}}(\boldsymbol{q}(t), \dot{\mathfrak{q}}(t), t, \Delta t, \dot{\mathfrak{q}}(t), \boldsymbol{p}, \dot{\mathfrak{x}}_{\text{des}}^{\text{base}})$ is an algorithm of the control policy used to control many of the quadrupedal robots in this thesis.

1: $\{\boldsymbol{x}(t), \dot{\mathfrak{x}}(t)\} \leftarrow$ FORWARDKINEMATICS$(\boldsymbol{q}(t), \dot{\mathfrak{q}}(t))$
2: $\text{mode} \leftarrow \{\texttt{STANCE}, \texttt{STANCE}, \texttt{STANCE}, \texttt{STANCE}\}$ ▷ mode_i determines the behavior of each foot i
3: **for each** $\text{foot} \in \{\text{foot}_{\{\text{left,front}\}}, \text{foot}_{\{\text{right,front}\}}, \text{foot}_{\{\text{left,hind}\}}, \text{foot}_{\{\text{right,hind}\}}\}$ **do**
4: $\quad \{\boldsymbol{x}_{\text{des}}^{\text{foot}}, \dot{\mathfrak{x}}_{\text{des}}^{\text{foot}}, \text{mode}\} =$ GAITPLANNER$(\dot{\mathfrak{x}}_{\{\text{base,des}\}}, \boldsymbol{q}, \text{mode}, \boldsymbol{p}, \text{foot}, t)$ ▷ see Algorithm 7
5: $\{\boldsymbol{q}_{\text{des}}(t + \Delta t), \dot{\mathfrak{q}}_{\text{des}}(t + \Delta t)\} \leftarrow$ INVERSEKINEMATICS$(\boldsymbol{q}(t), \boldsymbol{x}_{\text{des}}(t + \Delta t), \dot{\mathfrak{x}}_{\text{des}}(t + \Delta t))$
6: $\{\boldsymbol{u}(t)\} \leftarrow$ INVERSEDYNAMICS$(\boldsymbol{q}(t), \dot{\mathfrak{q}}(t), \dot{\mathfrak{q}}_{\text{des}}(t + \Delta t), \Delta t)$
7: **return** $\{\boldsymbol{u}(t)\}$

9.2 Gait Planning

This section describes the reactive planning system for online, dynamic locomotion trajectory generation and tracking. The quadrupedal robots in this thesis follow trajectories generated by an open source legged locomotion software PACER[14].

A gait is the pattern of movement that the limbs of an animal (or a robot) follow during locomotion. PACER implements a simple gait planner that generates desired motions for the feet of the robot from a set of gait or, more generally, control policy parameters, and a desired motion for the body (i.e., base, torso, root link) of the robot.

The commands generated by the planner are defined in operational space. The motion of a robot in stance phase is defined by a small set of "gait parameters" (see §9.2.1); all that determines the robot's motion plan during locomotion are these parameters, the mode of each foot (past states of Algorithm 7 in Section 9.2.2), the base velocity command (§9.4), and the robot's state and kinematics. Whether a foot is in swing or stance phase is determined by a gait timing pattern; how a gait timing is generated is described in Section 9.2.3. Stance foot behavior (§9.2.4) is determined by only the current base velocity command (i.e., the behavior maintains no history) and is calculated at each controller iteration. Swing phase behavior (§9.2.5) is calculated as a velocity-clamped cubic spline at the start of each swing phase and is re-planned during that swing phase if the robot's commanded velocity changes. The following sections describe how the gait planner in this section fits into the control policy for a locomoting robot (§9.3) and describes how

[14] PACER is available at `http://github.com/PositronicsLab/Pacer`

the base velocity commands are generated (§9.4).

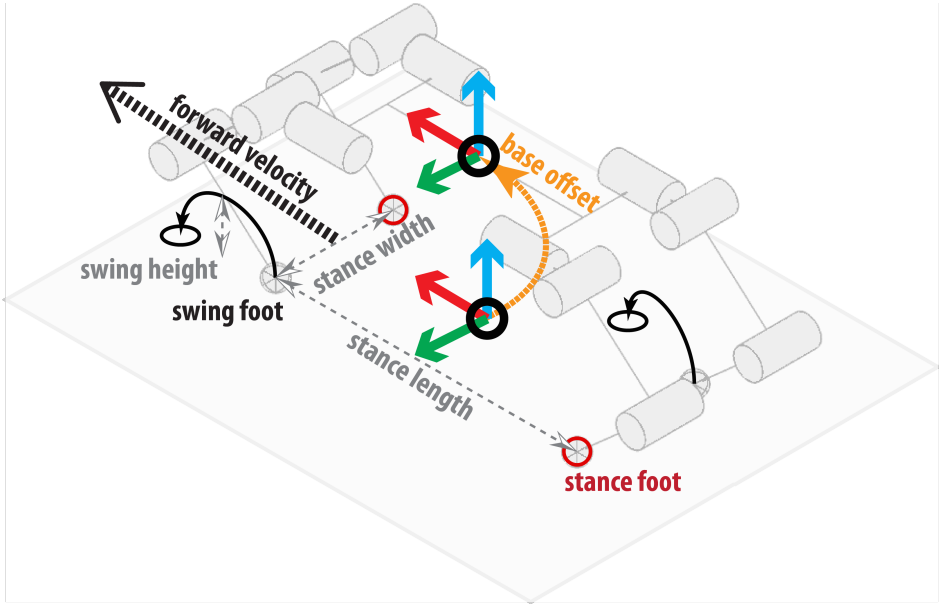


Figure 66: ***Visualization of the gait parameters for a quadrupedal gait.***

9.2.1 Gait parameters The gait for a quadrupedal robot is defined in PACER using a short list of parameters (see Table 17). Among quadrupedal mammals, there exists a relationship between some morphological traits and parameters of an optimal gait; these relationships are discussed in (Herr et al., 2002). Herr et al. (2002) describes that all of the gait parameters need not be explicitly defined, rather, many of them may be generated automatically with only the desired velocity of the robot left as input into a gait planner.

Gait Parameters

Parameter	description
desired velocity (max between waypoints)	$\dot{x}_{\{\text{base,des}\}} = \{\text{forward}, \text{strafe}, -, -, -, \text{rotation}\}$
base linear offset (absolute or relative to max reach)	{ x-offset, y-offset, height }
base orientation offset (rad)	$\{roll, pitch, yaw\}$
step height (cm)	the height of the two via points in the step trajectory
stance length (absolute or relative to base width)	the average distance between front and back stance feet
stance width (absolute or relative to base width)	the average distance between left and right stance feet
gait duration (sec)	interval of time where one gait cycle is performed
liftoff timing (% gait duration)	Point in the gait duration at which a stance phase ends of a certain foot and a flight phase begins for a duration of $100\% - \textit{duty factor}_i$ {*left front*, *right front*, *left hind*, *right hind*}
duty factor (% gait duration)	proportion of *gait duration* that each foot spends in stance phase over a full gait cycle. {*left front*, *right front*, *left hind*, *right hind*}

Table 17: *Gait parameters for input into the locomotion system.*

9.2.2 Gait planning algorithm The gait planner (illustrated in Figure 67 and described in Algorithm 7) takes as input desired planar base velocity ($\dot{x}_{\{\text{base,des}\}}$) and outputs the desired position and velocity of each foot to drive the robot across the environment at the desired velocity. The planner outputs a trajectory for each foot in the local frame of the robot. After end effector trajectories have been planned, joint trajectories are determined at each controller cycle using inverse kinematics.

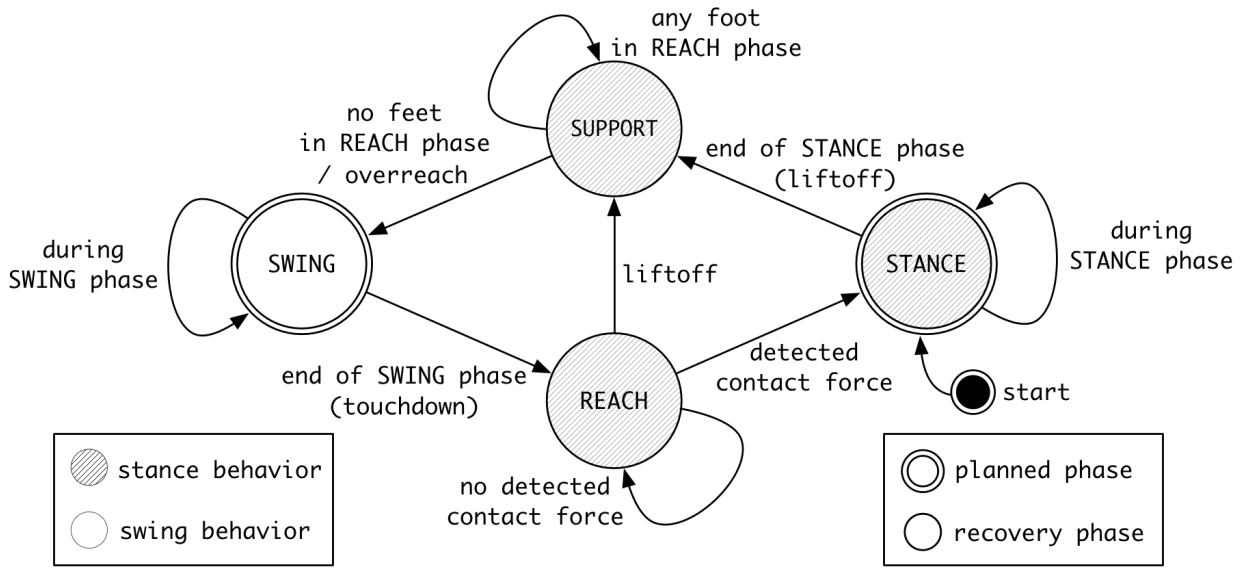


Figure 67: ***Locomotion planner flowchart depicting the mode switches for a single foot over the course of the gait. Double-outlined states are planned, while single-outlined states are recovery behaviors.***

During a stance phase a robot's foot will drive the robot forward until the next lift-off phase. The movement of the stance feet during locomotion must satisfy its velocity command using a specified gait; this means that foot placement must be carefully selected so that the stance foot will not leave the reachable region for that foot during the duration of a stance phase. Figure 67 adds two recovery phases to the standard stance and swing phase called "support" and "reach" that are active after a phase is supposed to end, but before it is safe to begin the next phase. The support phase continues after the end of a stance phase; it checks to make sure that the robot has not diverged from plan (i.e., a touchdown was planned but no contact was detected) before permitting a continuation of the gait. The reach step is a continuation of the swing phase; it activates as a recovery action of a touchdown was planned but no contact was detected; it commands the robot to push the stance foot that does not detect contact down toward where the ground should be. If a foot encounters the edge of the reachable space of the robot, an additional recovery action is triggered—starting a new swing phase to search for a new foothold.

Algorithm 7 $\{\boldsymbol{x}_{\text{des}}^{\text{foot}}, \dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}}, \text{mode}_{\text{foot}}\} = \text{GAITPLANNER}(\dot{\text{x}}_{\text{des}}^{\text{base}}, \boldsymbol{q}, \text{mode}, \boldsymbol{p}, \text{foot}, t)$ is a planning algorithm for determining the desired position $\boldsymbol{x}_{\text{des}}^{\text{foot}}$ and velocity $\dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}}$ of each foot with respect to the robot's base link frame (i.e., when the robot is standing: the x-axis is forward, the y-axis is left, and the z-axis is up). Figure 67 depicts this algorithm as a flow chart. The algorithm should always initialize with parameter $\text{mode} = \{\texttt{STANCE}, \texttt{STANCE}, \texttt{STANCE}, \texttt{STANCE}\}$. Time t must increase monotonically for this algorithm to function.

$\{\boldsymbol{x}_{\text{foot}}, -\} \leftarrow \text{FORWARDKINEMATICS}(\boldsymbol{q}, -)$
progress $\leftarrow t \mod \boldsymbol{p}_{\text{gait period}}$ ▷ current point in gait cycle
if progress $\in \boldsymbol{p}_{\text{stance phase interval(foot)}}$ **then** ▷ if in planned stance phase
  touchdown $\leftarrow (\text{mode}_{\text{foot}} \neq \texttt{STANCE})$ ▷ if planned stance phase just began, activate touchdown
else
  liftoff $\leftarrow (\text{mode}_{\text{foot}} \neq \texttt{SWING})$ ▷ if planned stance phase just ended, activate liftoff
if $\text{mode}_{\text{foot}} \in \{\texttt{STANCE}, \texttt{REACH}, \texttt{SUPPORT}\}$ **then** ▷ "stance phase" behavior (§9.2.4)
  $\boldsymbol{x}_{\text{des}}^{\text{foot}} = \boldsymbol{x}_{\text{foot}}$
  $\mathbf{J}_{\{\text{foot,base}\}} \leftarrow \text{JACOBIAN}_{\text{foot}}(\boldsymbol{q})$ ▷ Generate foot Jacobian (see §9.2.4)
  $\dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}} \leftarrow -\mathbf{J}_{\{\text{foot,base}\}}\dot{\text{x}}_{\text{des}}^{\text{base}}$ ▷ Push robot base forward at goal velocity (see §9.2.4)
  if $\text{mode}_{\text{foot}} = \texttt{REACH}$ **then** ▷ reach recovery phase
    if CONTACTSENSOR(foot) **then** ▷ Foot sensors detect sufficient force to support a stance phase
      $\text{mode}_{\text{foot}} \leftarrow \texttt{STANCE}$
    else
      $\dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}} \leftarrow \dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}} - ||\dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}}||\hat{\boldsymbol{z}}$ ▷ move foot downward in operational space
  if liftoff **then** ▷ planned end of stance phase
    $\text{mode}_{\text{foot}} \leftarrow \texttt{SUPPORT}$ ▷ wait for sufficient support to begin a swing phase with foot
    $t_{\texttt{SUPPORT}} \leftarrow t$ ▷ record when support phase began
  if $\text{mode}_{\text{foot}} = \texttt{SUPPORT}$ **then** ▷ fail-safe stance phase
    if $\forall_{\text{feet}}$, $\text{mode} \neq \texttt{REACH}$ **then** ▷ the robot is not in a recovery mode with another foot
      $\text{mode}_{\text{foot}} \leftarrow \texttt{SWING}$ ▷ start a flight phase to next foothold
    else if $||\boldsymbol{x}_{\text{foot}}|| > \texttt{max\_reach}$ **then** ▷ The foot has pushed beyond maximum reach (overreach)
      $\text{mode}_{\text{foot}} \leftarrow \texttt{SWING}$ ▷ forcibly proceed to swing phase
if $\text{mode}_{\text{foot}} = \texttt{SWING}$ **then** ▷ "swing phase" behavior (§9.2.5)
  if liftoff **then** ▷ swing phase has just begun
    $t_{\texttt{SWING}} \leftarrow t$ ▷ record when swing phase began
    $t_{\text{touchdown}} \leftarrow t_{\texttt{SWING}} + \boldsymbol{p}_{\text{gait period}}(1 - \boldsymbol{p}_{\text{duty factor(foot)}})$ ▷ get time of planned touchdown
    $t_{\text{touchdown}} \leftarrow t_{\text{touchdown}} - (t_{\texttt{SWING}} - t_{\texttt{SUPPORT}})$ ▷ reduce swing phase by duration of support phase
  $t_{\text{swing duration}} \leftarrow t_{\texttt{touchdown}} - t$ ▷ duration over which swing spline will be valid
  $\mathbf{T}_{\text{foot}} \leftarrow \text{SWINGPLAN}(\dot{\text{x}}_{\text{des}}^{\text{base}}, \boldsymbol{x}_{\text{foot}}, \dot{\boldsymbol{x}}_{\text{foot}}, \boldsymbol{q}, t_{\text{swing duration}})$ ▷ calculate swing spline (see Figure 71)
  $\{\boldsymbol{x}_{\text{des}}^{\text{foot}}, \dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}}\} \leftarrow \mathbf{T}_{\text{foot}}(t - t_{\texttt{SWING}})$ ▷ evaluate spline to get swing foot behavior
  if touchdown **then** ▷ planned end of swing phase
    $\text{mode}_{\text{foot}} \leftarrow \texttt{REACH}$ ▷ begin reach recovery phase
return $\{\boldsymbol{x}_{\text{des}}^{\text{foot}}, \dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}}\}$

9.2.3 Gait Timing Gait timing involves choosing when each foot is supporting the robot and when it is moving to a new foothold. Gaits can be challenging to generate from a reduced set of parameters because the parameters must coordinated in such a way that the resulting motion does not simply topple the robot. The locomotion system described in this chapter affects stability by

carefully adjusting when touchdown and liftoff occur. The only systems stabilizing the robot are the reactive "support" and "reach" phases of the planner and the implicit stabilizing effect that a gait timing imparts to a robot's dynamics. Gait parameterization is thus a crucial aspect to ensuring that such failing or unstable configurations are rare in the parameter space of a gait. The following terms are used to describe the various timed actions of a gait planning system:

stance phase The interval of time where a foot is planned to be in contact with the ground (foot applies force to move the robot)

swing phase The interval of time where a foot is planned to not be in contact with the ground (to position the foot for the next stance phase)

flight phase An interval of time where no feet are planned to be in contact with the ground (no feet in stance phase, all feet in swing phase). A gait that exhibits a flight phase is typically referred to as a "run" or a "flighted gait"; A gait that does not exhibit a flight phase is typically referred to as a "walk".

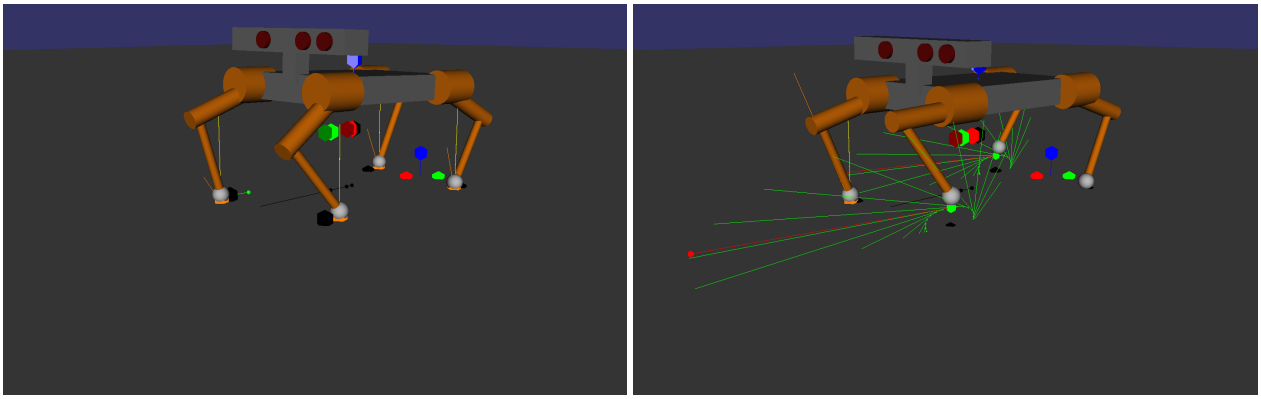
duty-factor the duration of the stance phase for a foot as a proportion of the gait period duration

touchdown The moment when a foot is planned to make contact with the ground and transition from swing to stance phase

liftoff The moment when a foot is planned to break contact with the ground and transition from stance to swing phase

gait pattern A data structure combining the liftoff and touchdown times for all feet of the robot.

gait period The duration of the interval of time over which a cyclic gait pattern is repeated.



(a) ***Pre-step*** **(b)** ***During step***

Figure 68: ***two frames of a straight-forward step using the*** PACER ***locomotion system. The left image depicts all feet in stance phase and the right image depicts the left front an right hind feet in swing phase. Debugging visualization information in the images include contact normal, contact force vector, swing foot trajectories, base link frame, global frame, expected location of the robot at touchdown, and each foot "origin" (i.e., neighborhood around which a foot is expected to operate).***

In PACER only a single parameter per foot of the robot ("touchdown") must be adjusted in order to produce a new gait for the robot. A few typical touchdown and liftoff timings (gait patterns) are depicted in Figure 69.

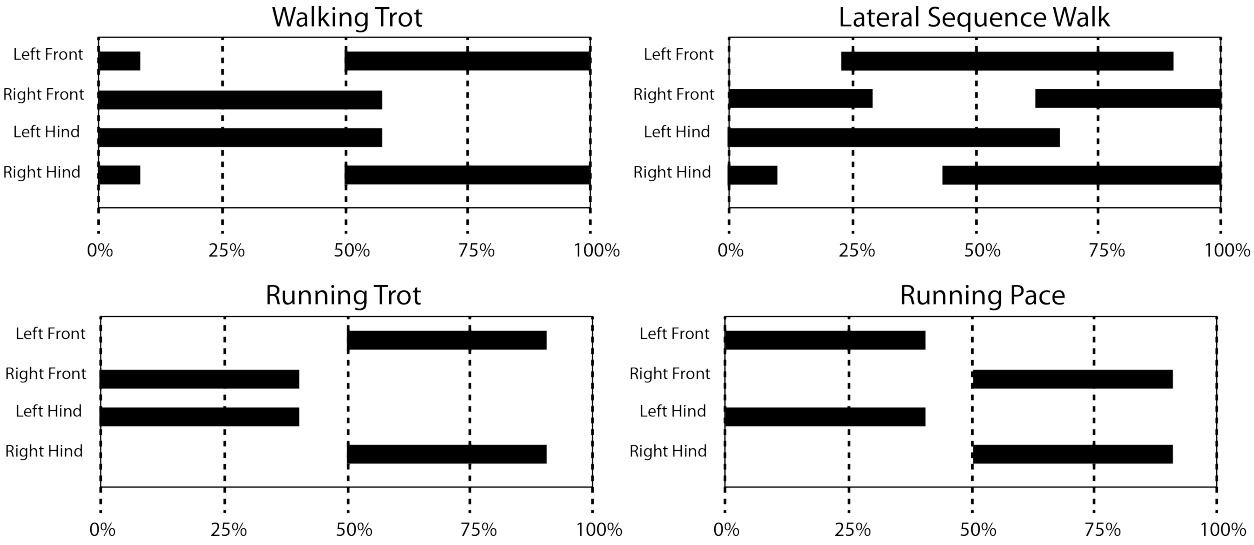


Figure 69: ***Plots of timings for various quadrupedal gaits. Darkened bars indicate a stance phase and empty regions indicate a swing phase. The percent values on the horizontal axis refer to the progress in the gait through the total duration of the gait period. Converting these plots to gait parameter values in Table 17: The "walking trot" has touchdown times*** $\{50\%, 0\%, 0\%, 50\%\}$***, duty factors for all feet equal to*** 60%***, and liftoff times*** $\{10\%, 60\%, 60\%, 10\%\}$**.**

Gait timings are typically copied from nature after careful observation of quadrupedal animals in motion. The search space for new gaits exhibits a complex interdependence between gait timing, desired speed of locomotion, and the morphology of the locomoting system; rather than discover new, functional gait timings, the typical approach is to select one of the many known gaits and then make small adjustments to it in order to fit a desired task. For example, slow movement (walk) might typically be paired with a lateral sequence walk, amble, or walking trot while a faster movement might require a running trot, pace, or gallop.

9.2.4 Stance Phase A robot propels itself forward during the stance phase of its gait (see Section 9.2.4). Forward propulsion is achieved in legged locomotion by the feet pushing back against the ground to push the robot forward.

The recovery actions in Figure 67 are mostly applicable to walking gaits, as running gaits are too dynamic for large, online adjustments to the timing and kinematic properties of a gait while also ensuring stability. Instead, running gaits can use contact forces at the feet to adjust the momentum of the robot over the duration of a stance phase, toward promoting stability. Figure 70 depicts the use of a stance phase to propel a robot at a constant horizontal velocity.

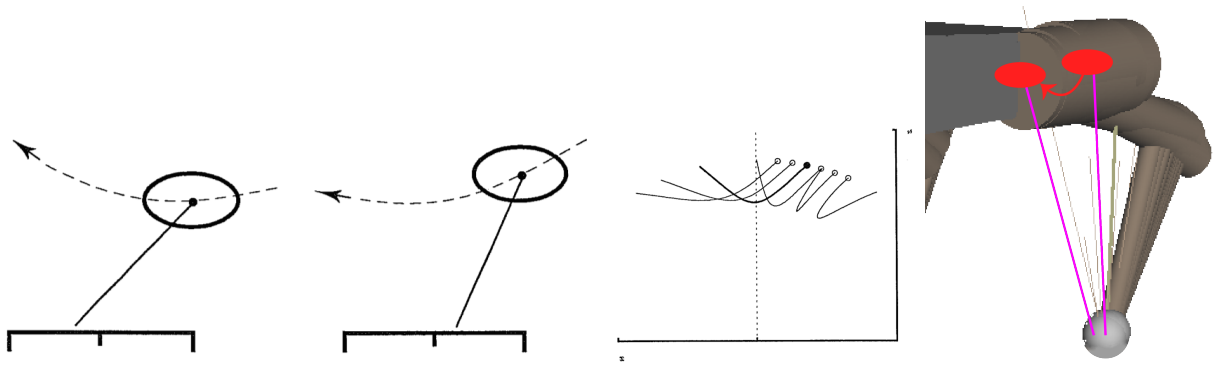


Figure 70: ***Catch points: foot placement determines the moment a foothold exerts on the majority of the mass of a robot. The selection of a foothold determines the profile of the ground reaction forces over the duration of a support phase. A foothold is selected that is centered about the midpoint in the stance phase, leading to a symmetric force profile in the robot's sagittal plane. Images are copied from Raibert (1986), Chapter 2: Hopping on One Leg in a Plane***

The stance foot crosses under the point of support of the limb at about mid-stride in order to have

equal loading and unloading phases of the stance phase to maintain a constant speed. Figures 70a and 70b show how this applies to monopods; the locomotion system has applied this principle to each leg of the robot (Figure 70c).

Definition of the foot-base Jacobian In order to determine what direction a foot should push against the environment in order to induce a desired motion of the base, a foot-base Jacobian is calculated by finite differencing or is analytically derived from the robot's kinematics. The foot-base Jacobian $\mathbf{J}_{\{\text{foot,base}\}}$ is a block of the Jacobian $\mathbf{J}_{\{\text{operational,configuration}\}}$ relating the effect that changes to a robot's generalized coordinates have on the coordinates of a robot's links in operational space; this matrix is the partial derivative $\frac{\partial f}{\partial \boldsymbol{q}}$ of the function $\boldsymbol{x} = f(\boldsymbol{q})$, which relates the position $\boldsymbol{x}$ of some point on one of the robot's links (in this case, the point of contact between the foot and the ground) to the robot's generalized coordinates in configuration space $\boldsymbol{q}$. The transpose of $\mathbf{J}_{\{\text{operational,configuration}\}}$ is also known to map forces applied to a point on a robot's link to generalized forces in configuration space. The matrix used by the gait planner $\mathbf{J}_{\{\text{foot,base}\}} \in \mathbb{R}^{3\times 6}$ is the block of $\mathbf{J}_{\{\text{operational,configuration}\}} \in \mathbb{R}^{(3\cdot nc)\times(nq+6)}$ which maps from the generalized coordinates of the robot's base in configuration space ($6d$) to the position of the point on a stance foot where it contacts the environment ($3d$), where nc is the number of stance feet and nq is the number of actuated joints on the robot.

The gait planner utilizes the relationship between the velocity of the point on a stance foot where it contacts the environment and the planned velocity of the robot base to determine the planned velocity of a foot; matrix $\mathbf{J}_{\{\text{foot,base}\}}$ transforms between these two velocities

$$\dot{\boldsymbol{x}}_{\text{des}}^{\text{foot}} \leftarrow -\mathbf{J}_{\{\text{foot,base}\}}\dot{\boldsymbol{x}}_{\{\text{base,des}\}}$$

The value in Equation 9.2.4 is negated because the foot is being driven to push the robot base at the desired motion $\dot{\boldsymbol{x}}_{\{\text{base,des}\}}$. The value $\mathbf{J}_{\{\text{foot,base}\}}\dot{\boldsymbol{x}}_{\{\text{base,des}\}}$ would evaluate to: *the velocity of a point on the foot link if the robot's joints were frozen in place and the base were moving at the desired velocity*; instead, the transformation required to command the robot to walk forward is the

opposite of this: *the joints of the robot are moving to push the robot base into motion*.

9.2.5 Swing Phase The purpose of the swing phase is to reposition the foot to a foothold that will be within reach of the robot and provide balancing support for the robot over the course of the next stance phase. The foot follows a planned path through the air over the duration of the stance phase in order to reach a foothold at touchdown, the moment that the swing phase is planned to end (calculation of the touchdown position is shown in Section 9.2.5). The planned path for a swing foot is generated by calculating a four knot (i.e., control point, via point) cubic spline with velocity constraints at each end point. The spline is then evaluated with respect to the time since the swing phase started over the course of the step. Figure 71 illustrates how each of the four swing phase control points are determined.

1. The first "liftoff" control point is set to the current foot velocity at the current foot position
2. The second via point is set to $\mathrm{step\_height}$ above the liftoff position
3. The third via point is set to $\mathrm{step\_height}$ above the touchdown position with an additional horizontal displacement equal to $\mathrm{overshoot} \times (\boldsymbol{x}_{\mathrm{touchdown}} - \boldsymbol{x}_{\mathrm{liftoff}})$ beyond the touchdown location.
4. The fourth and last "touchdown" control point is set to the touchdown foot position and the desired foot velocity at the time of touchdown (see §9.2.5).

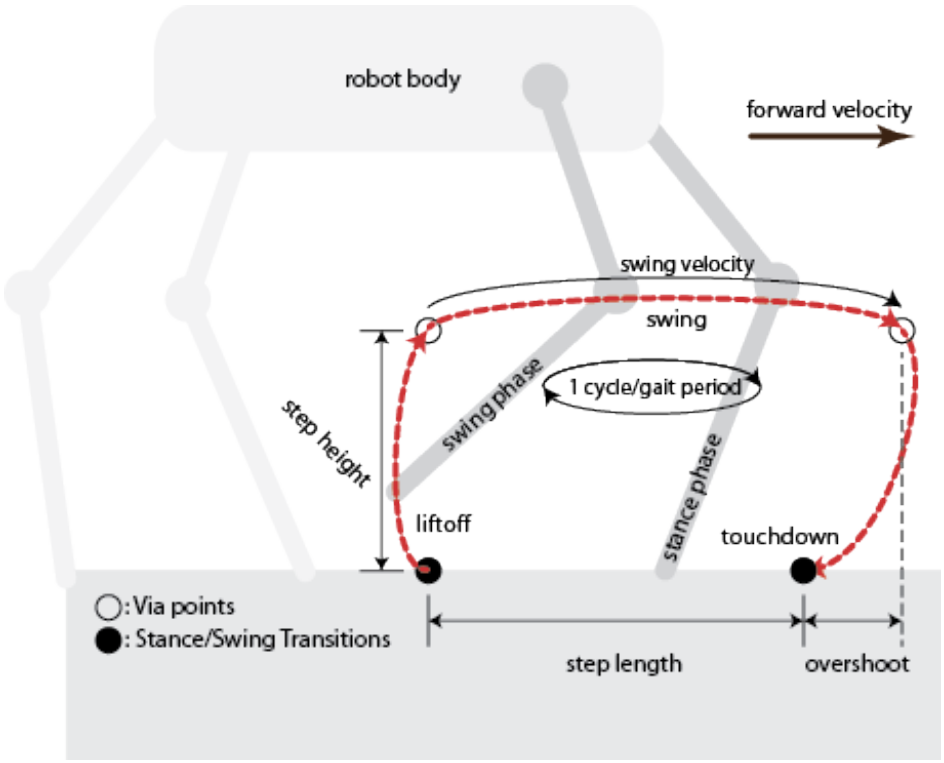


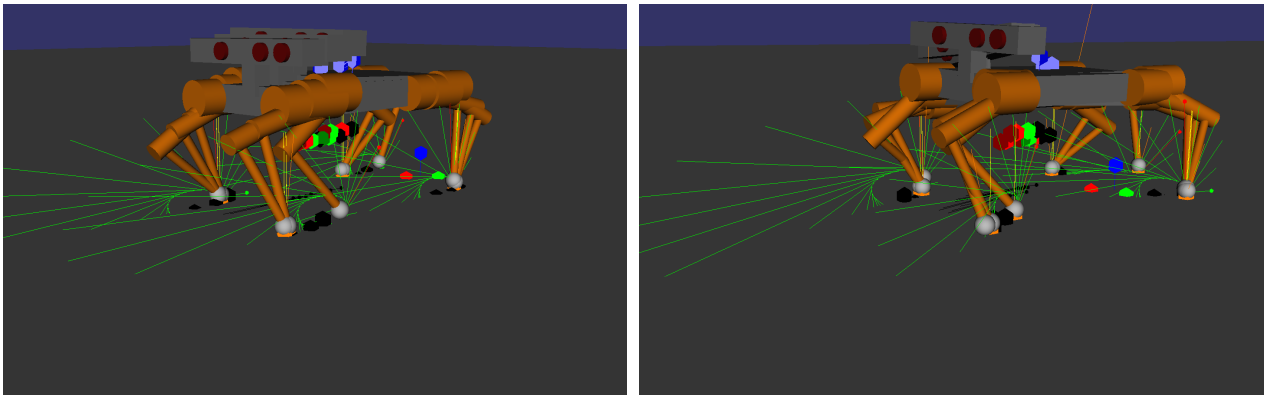
Figure 71: ***Visualization of gait parameters of a quadrupedal gait. This this diagram labels a projection of the same gait planning system in swing phase onto a quadruped's sagittal plane.***

Touchdown foot placement The quarupedal robot affects a monopod locomotion strategy with each leg by predicting where the base of the robot will be both at the moment of touch down and at the middle of its next stance phase for each foot. The future states of the robot are represented as a first-order differential equation:

$$\dot{x}_{\text{base}}(t) = f(t, y(t)), \quad y(t_0) = \boldsymbol{x}_{\text{base}}(t_0)$$

This equation is integrated over interval $[t_0, t_f]$ to calculate the position of the robot base at time t_f ($\hat{\boldsymbol{x}}_{\text{base}}(t_f) = y(t_f)$). Following from the strategy used for straight-line motion, turning is trivial to implement (see Figure 72b). Foot placement is only based upon the predicted location of the base, which follows a planar process model. The difference between the robot's state at its next touchdown and half-way through the next stance phase determines where the robot will place its foot. Foot placements are generated with the intention of having each of the robot's shoulders

directly above its foot at the mid-point of that foot's stance phase.



(a) ***Foot placement during forward walking*** **(b)** ***Foot placement during turning***

Figure 72: ***The robot predicts foot placement based on where it will be during its next stance phase if controlled at a constant velocity*** $\dot{x}_{\{\text{base,des}\}}$***. This strategy is used to determine the touchdown point of each swing phase (*** $\mathbf{T}_{\text{foot},n}$, where $n = |\mathbf{T}|$***)***

Figure 72 demonstrates both of the mentioned mechanics at work. (1) The touchdown point of the front left foot is predicted so that the robot's shoulder will pass over it mid-way through the next stance phase. (2) The robot's left hind hip dips during loading and then rises during unloading as the robot progresses through the stance phase. This induced vertical oscillation from the motion of (2) allows the robot to transfer its weight between stance phases by reducing the robot-ground reaction forces during unloading—due to the upward motion of the robot.

9.3 Plugin-based robot interface and control architecture

PACER maintains and updates an internal model of the robot being controlled; it parses the inertial and dynamic properties from standard robot description file types (SDF, URDF) and uses this model to generate kinematic (Jacobian), and dynamic (generalized inertia tensor, momentum) data independently of any particular simulator's kinematics and dynamics representations. This independence from a particular simulator allows for ready communication between groups that conduct research within different simulators. The locomotion system has been demonstrated controlling real and simulated robots with no alteration to the underlying controllers (see Figure 73).

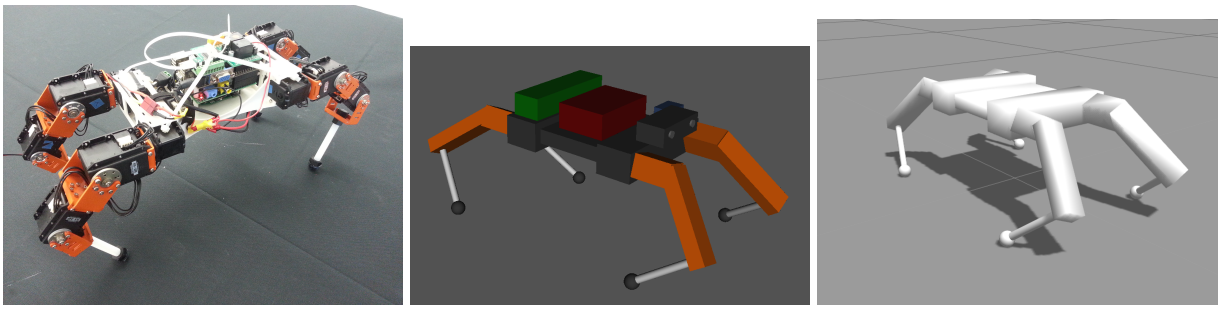


Figure 73: ***The quadruped robot*** **R. Links** ***(left), in*** **MOBY** ***(center), and*** **GAZEBO** ***(right).***

9.3.1 Modular planning and control framework The control architecture of PACER adopts a plugin framework; such a framework is also used to manage robot controllers in GAZEBO (Koenig & Howard, 2004). PACER modularizes each controller and planner into its own encapsulated unit. Though some systems may bridge planning and control into a single system (kinodynamic planning), the active systems on a robot often are segmented into distinct, sequential categories: perception, planning, and control. However, one may use further categorizations, including global and local planning, reactive control, stabilization, balance, inverse kinematics, inverse dynamics control, and error feedback control. PACER ensures that sequential processes (*e.g.*, footstep planning then inverse kinematics) are run in order, while allowing non-interdependent processes—such as global path planning—to run in parallel. Plugin scheduling is further extended by adding a real-time factor to each control module, permitting a controller to run for n iterations of the real time system before it is queried for a control input to the robot. This layered architecture is illustrated in Figure 74.

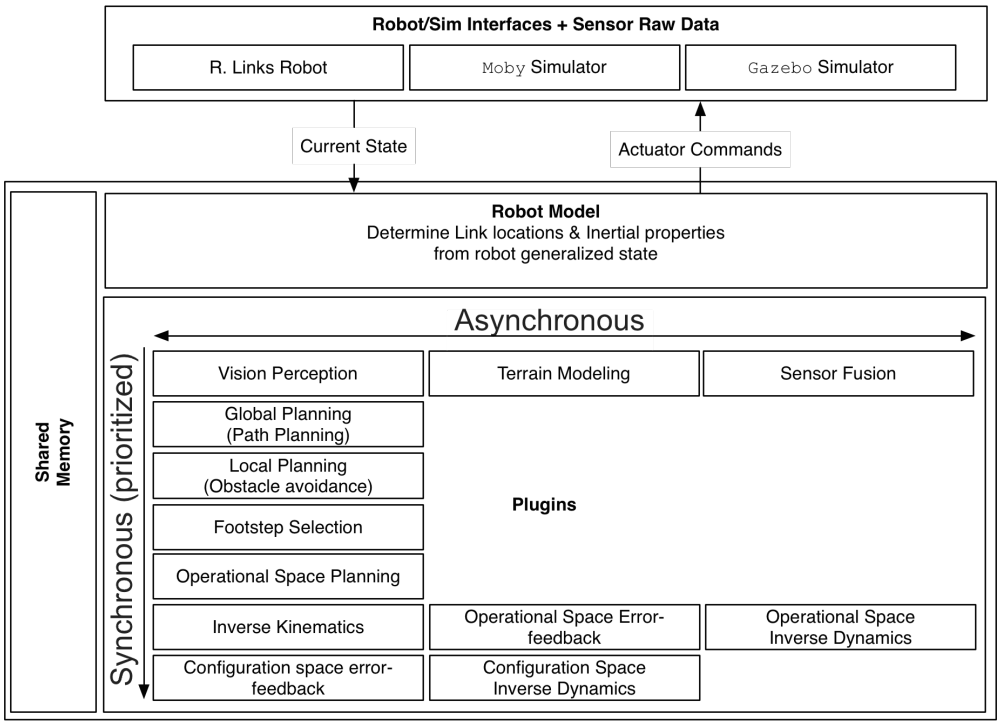


Figure 74: ***A flow-chart of planning an control data as it passes through the standard set of* PACER *plugins used by a quadrupedal robot.***

9.4 Driving and Navigation

PACER provides a simple front end to semi-autonomous control in locomotion by abstracting legged robot footstep planning and control to more simplistic commands (*e.g.*, planar robot degrees of freedom (x, y, θ), the base position differential of the robot (in $SE(2)$). Using this abstraction, an operator can drive a robot like a car, or have the robot base follow a planned trajectory between waypoints in the environment.

9.4.1 Steering There are two types of movement typical robot's might use while "driving", holonomic (planar) movement, and a specific class of non-holonomic movement that corresponds to dual-axle steering car such as a Dubin's car model (see Figure 75). An operator will likely be familiar with both models, each pertaining to a different task; a sprinting robot might only make small steering adjustments but avoid side-stepping (non-holonomic); a slow moving robot might strafe sideways for a small position adjustment rather than turn, more forward, and turn back. While holonomic planar movement is extremely useful for problems that are particularly hard for typical

car-like controls, such as parallel parking and turning in place, it is useful to keep a locomoting robot facing forward. When a robot's forward axis is tangent to the path it is following sensory equipment on the head might be better oriented in the direction of movement. Forward-facing motion also helps keep the body of h robot oriented to a locally-defined "forward" frame, which might simplify stabilization (e.g., roll and pitch are can be readily identified in this context). Enabling control functionality using both of these potential steering behaviors would be beneficial, as each has a useful application in legged locomotion.

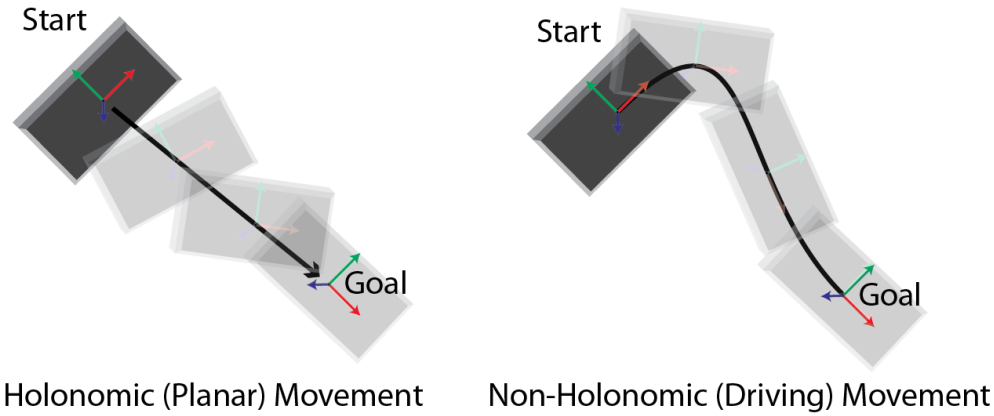


Figure 75: ***holonomic (planar) movement [left], and non-holonomic (driving) movement [right].***

9.4.2 Gamepad input PACER condenses complete control over most aspects of quadruped locomotion into a gamepad form-factor; the gamepad input consists of fourteen buttons and four continuous axes for fine-tuned control adjustments (see Figure 76).

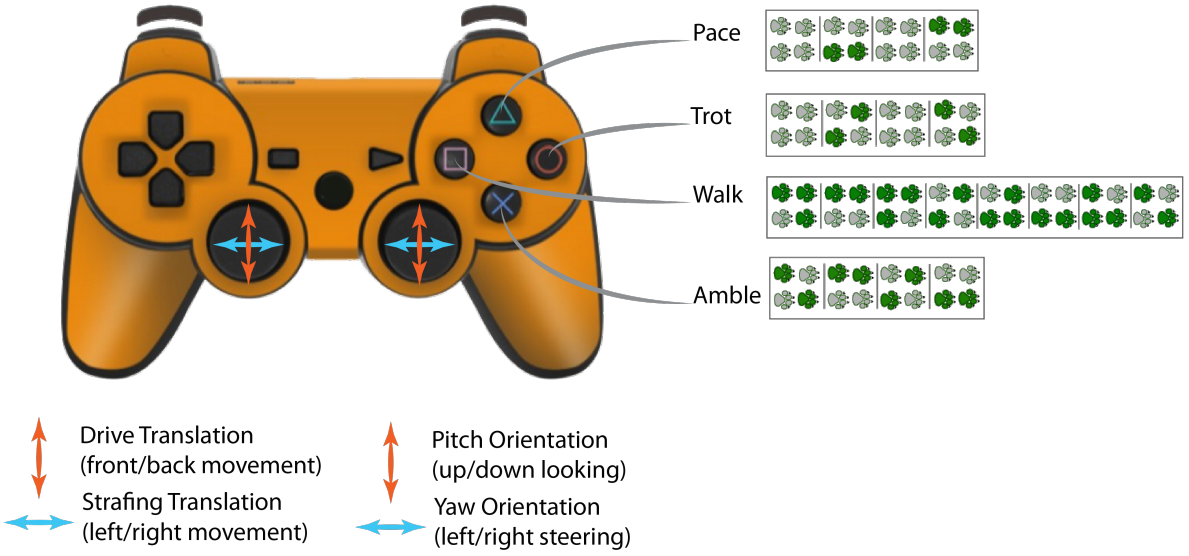


Figure 76: ***A view of the basic gamepad controls for semi-autonomous quadrupedal locomotion.***

The gamepad input layout in Figures 76 and 77 is an attempt at giving an operator full control over which gaits are used by the robot and how it moves through the world. The operator is prevented from producing failing gaits by having limited control over touchdown timing; there is a time gap between when a new gait is selected and when it is transitioned to by the robot (to perform a sanity check on the new gait an when to make the transition). Section 4.3.3 describes how such a sanity check is implemented through virtual falsification. These features achieve a semi-autonomous sharing of control between decisions made by the operator and reasoning made about the stability of a new control parameterization made by simulation.

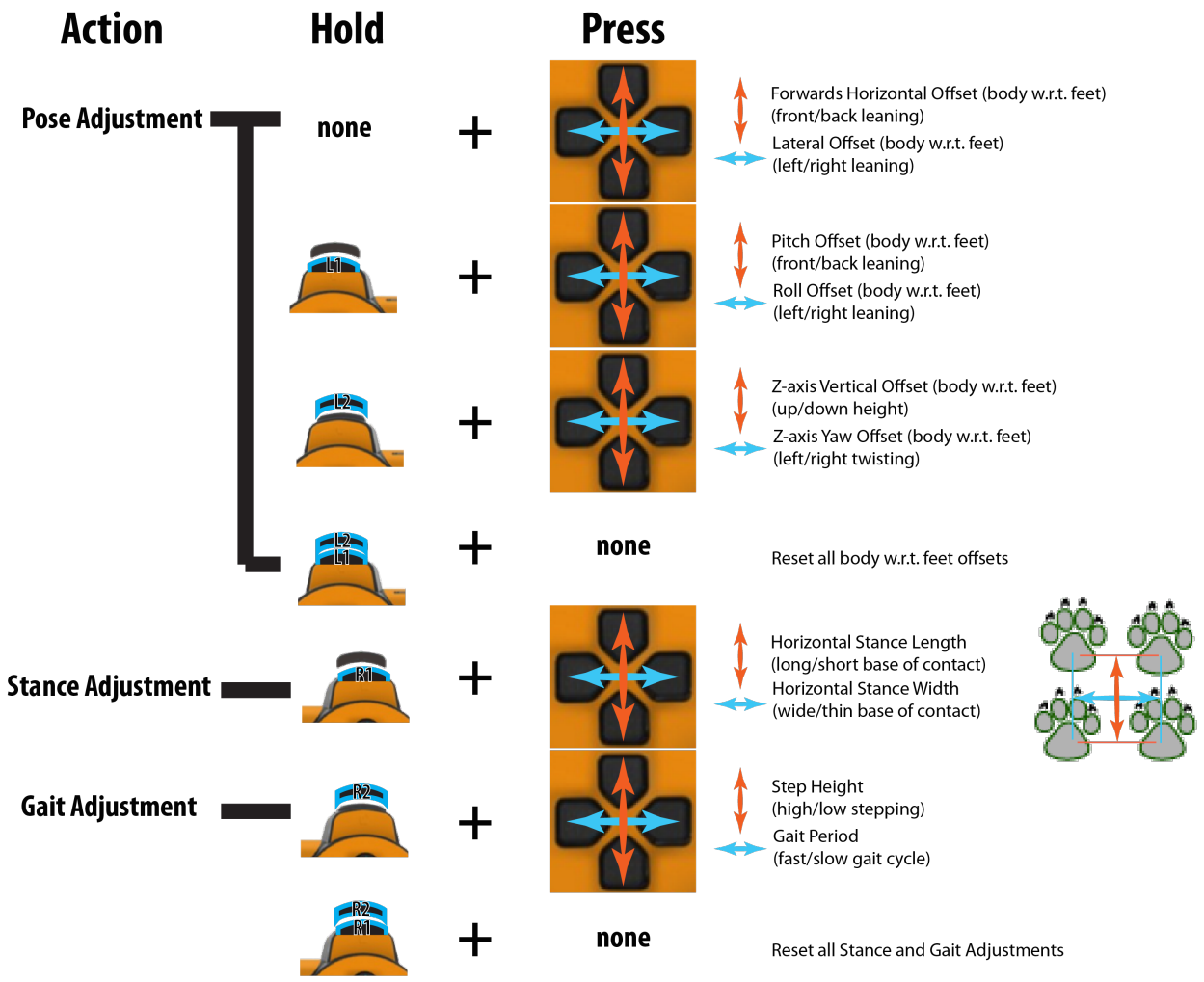


Figure 77: ***A view of the gamepad controls and mode toggles for advanced gait settings; these enable control over all relevant aspects of quadrupedal locomotion.***

9.5 Conclusion

Integrating robotic systems for planning, estimation, and control is a widely interdisciplinary task, drawing on the expertise of hardware engineers, software developers, and control theorists (among others). Researchers often have to collaborate across multiple institutions to create fully functional robots. The ease with which roboticists can communicate by passing techniques and software between research groups is a significant determining factor for how quickly robot systems can be prototyped, built, and controlled. Cooperation and standardization help to assess one's

work with respect to the state-of-the-art, and permit effective adoption and development upon existing research in the field. PACER provides a code base for locomotion. It implements a modular control architecture that enables researchers to contribute directly to the state of the art in locomotion, while permitting them to easily appraise, modify, and adopt existing software. PACER was developed with the aforementioned goals in mind.

10 Discussion, Conclusions, and Future Work

The convergence toward an effective robot through trial and error, simulated or *in situ*, is tedious and time consuming. The reward for following this approach is that it gives the few roboticists closely involved a repertoire of experiential knowledge that they can use to develop better systems in the future. This thesis described an extension to typical methods of robot prototyping and testing that seeks to bypass some of the physical experimentation currently used when developing a robot that performs well *in situ*. My aim has been to greatly accelerate the design-build-test cycle of research in robotics by providing a readily usable virtual testing and design framework for robot hardware and software. Toward that goal, this dissertation presented methods for automating and or simplifying a roboticist's typical workflow (i.e., designing, testing, controlling, and debugging robots). This dissertation described tools for fast, automated testing and interactive robot design that will give new roboticists a lower barrier to entry (in both dollars and time) for developing new robotic systems and will provide experienced roboticists with focused computer-aided engineering tools to help optimize workflows.

Appendix

Appendix A: Generalized contact wrenches

A contact wrench applied to a rigid body will take the form:

$$\boldsymbol{q} \equiv \begin{bmatrix} \hat{\boldsymbol{q}} \\ \boldsymbol{r} \times \hat{\boldsymbol{q}} \end{bmatrix} \tag{156}$$

where $\hat{\boldsymbol{q}}$ is a vector in $\mathbb{R}^3$ and $\boldsymbol{r}$ is the vector from the center of mass of the rigid body to the point of contact (which is denoted $\boldsymbol{p}$). For a multi-rigid body defined in m minimal coordinates, a *generalized contact wrench* $\mathbf{Q} \in \mathbb{R}^m$ for single point of contact $\boldsymbol{p}$ would take the form:

$$\mathbf{Q} = \mathbf{J}^{\mathsf{T}} \boldsymbol{q} \tag{157}$$

where $\mathbf{J} \in \mathbb{R}^{6 \times m}$ is the manipulator Jacobian (see, *e.g.*, Sciavicco & Siciliano, 2000) computed with respect to $\boldsymbol{p}$.

Appendix B: The Principal Pivoting Method for solving LCPs

The Principal Pivot Method I (Cottle, 1968; Murty, 1988) (PPM), which solves LCPs with P-matrices (complex square matrices with fully non-negative principal minors (Murty, 1988) that includes positive semi-definite matrices as a proper subset). The resulting algorithm limits the size of matrix solves and multiplications.

The PPM uses sets α, $\overline{\alpha}$, β, and $\overline{\beta}$ for LCP variables $\boldsymbol{z}$ and $\boldsymbol{w}$. The first two sets correspond to the $\boldsymbol{z}$ variables while the latter two correspond to the $\boldsymbol{w}$ variables. The sets have the following properties for an LCP of order n:

- $\alpha \cup \overline{\alpha} = \{1, \ldots, n\}$
- $\alpha \cap \overline{\alpha} = \emptyset$
- $\beta \cup \overline{\beta} = \{1, \ldots, n\}$

- $\beta \cap \overline{\beta} = \emptyset$

Of a pair of LCP variables, (z_i, w_i), index i will either be in α and $\overline{\beta}$ or β and $\overline{\alpha}$. If an index belongs to α or β, the variable is a *basic variable*; otherwise, it is a *non-basic variable*. Using this set, partition the LCP matrices and vectors as shown below:

$$\begin{bmatrix} \boldsymbol{w}_\beta \\ \boldsymbol{w}_{\overline{\beta}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\beta\alpha} & \mathbf{A}_{\beta\overline{\alpha}} \\ \mathbf{A}_{\overline{\beta}\alpha} & \mathbf{A}_{\overline{\beta}\overline{\alpha}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_\alpha \\ \boldsymbol{z}_{\overline{\alpha}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{q}_\beta \\ \boldsymbol{q}_{\overline{\beta}} \end{bmatrix}$$

Isolating the basic and non-basic variables on different sides yields:

$$\begin{bmatrix} \boldsymbol{z}_{\overline{\alpha}} \\ \boldsymbol{w}_{\overline{\beta}} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}_{\beta\overline{\alpha}}\mathbf{A}_{\beta\alpha} & \mathbf{A}_{\beta\overline{\alpha}}^{-1} \\ \mathbf{A}_{\overline{\beta}\alpha} - \mathbf{A}_{\overline{\beta}\overline{\alpha}}\mathbf{A}_{\beta\overline{\alpha}}^{-1}\mathbf{A}_{\beta\alpha} & \mathbf{A}_{\overline{\beta}\overline{\alpha}}\mathbf{A}_{\beta\overline{\alpha}}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_\alpha \\ \boldsymbol{w}_\beta \end{bmatrix} + \ldots$$
$$\begin{bmatrix} -\mathbf{A}_{\beta\overline{\alpha}}^{-1}\boldsymbol{q}_\beta \\ -\mathbf{A}_{\overline{\beta}\overline{\alpha}}\mathbf{A}_{\beta\overline{\alpha}}^{-1}\boldsymbol{q}_\beta + \boldsymbol{q}_{\overline{\beta}} \end{bmatrix}$$

If the values of the basic variables are set to zero, then solving for the values of the non-basic variables $\boldsymbol{z}_{\overline{\alpha}}$ and $\boldsymbol{w}_{\overline{\beta}}$ entails only computing the vector (repeated from above):

$$\begin{bmatrix} -\mathbf{A}_{\beta\overline{\alpha}}^{-1}\boldsymbol{q}_\beta \\ -\mathbf{A}_{\overline{\beta}\overline{\alpha}}\mathbf{A}_{\beta\overline{\alpha}}^{-1}\boldsymbol{q}_\beta + \boldsymbol{q}_{\overline{\beta}} \end{bmatrix} \tag{158}$$

PPM I operates in the following manner: (1) Find an index i of a basic variable x_i (where x_i is either w_i or z_i, depending which of the two is basic) such that $x_i < 0$; (2) swap the variables between basic and non-basic sets for index i (*e.g.*, if w_i is basic and z_i is non-basic, make w_i non-basic and z_i basic); (3) determine new values of $\boldsymbol{z}$ and $\boldsymbol{w}$; (4) repeat (1)–(3) until no basic variable has a negative value.

Appendix C: Proof that removing linearly dependent equality constraints from the MLCP in Section 7.3.4 does not alter the MLCP solution

Theorem 10.1 *The solution to the MLCP in Equations 74 and 75 without linearly dependent equality constraints removed from* $\mathbf{A}$ *is identical to the solution to the MLCP with reduced* $\mathbf{A}$ *matrix and unconstrained variables set to zero that correspond to the linearly dependent equality constraints.*

Proof Assume that $\mathbf{U}$ is a matrix with rows consisting of a set of linearly independent vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$, where $n \in \mathbb{N}$. Each of these vectors comes from a row of $\mathbf{P}$, $\mathbf{S}$, or $\mathbf{T}$. Assume $\mathbf{W}$ is a matrix with rows consisting of vectors $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m\}$, each of which is a linear combination of the rows of $\mathbf{U}$, for $m \in \mathbb{N}$. $\mathbf{U}$ and $\mathbf{W}$ are related in the following way: $\mathbf{Z} \cdot \mathbf{U} = \mathbf{W}$, for some matrix $\mathbf{Z}$. The MLCP from Equations 74 and 75 can then be rewritten as:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{U}^\mathsf{T} & -(\mathbf{ZU})^\mathsf{T} & -\mathbf{N}^\mathsf{T} \\ \mathbf{U} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{ZU} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{+}\boldsymbol{v} \\ \boldsymbol{f}_U \\ \mathbf{0} \\ \boldsymbol{f}_N \end{bmatrix} + \begin{bmatrix} \boldsymbol{\kappa} \\ \mathbf{0} \\ \mathbf{0} \\ \frac{^{-}\boldsymbol{\phi}}{\Delta t} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{w}_N \end{bmatrix} \tag{159}$$

$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{w}_N \geq \mathbf{0}, \boldsymbol{f}_N^\mathsf{T} \boldsymbol{w}_N = 0 \tag{160}$$

where $\boldsymbol{f}_U$ are unconstrained variables that correspond to the linearly independent equality constraints. Note that the value of $\mathbf{0}$ is assigned to the variables corresponding to the linearly dependent equality constraints. Since values for ${}^{+}\boldsymbol{v}$, $\boldsymbol{f}_U$, and $\boldsymbol{f}_N$ that satisfy the equations above require $\mathbf{U}\,{}^{+}\boldsymbol{v} = \mathbf{0}$, the constraint $\mathbf{ZU}\,{}^{+}\boldsymbol{v} = \mathbf{0}$ is automatically satisfied. ∎

Appendix D: Proof that no more than m positive force magnitudes need be applied along contact normals to a m degree of freedom multi-body to solve contact model constraints

This proof will use the matrix of generalized contact wrenches, $\mathbf{N} \in \mathbb{R}^{n \times m}$ (introduced in Section 7.3.1), and $\mathbf{M} \in \mathbb{R}^{m \times m}$, the generalized inertia matrix for the multi-body. $\boldsymbol{z}_I$ is the vector of contact force magnitudes and consists of strictly positive values.

Assume the rows of $\mathbf{N}$ are permuted and partitioned into r linearly independent and $n-r$ linearly

dependent rows, denoted by indices I and D, respectively, as follows:

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}_I \\ \mathbf{N}_D \end{bmatrix} \tag{161}$$

Then the LCP vectors $\boldsymbol{q} = \mathbf{N}\boldsymbol{v}$, $\boldsymbol{z} \in \mathbb{R}^n$, and $\boldsymbol{w} \in \mathbb{R}^n$ and LCP matrix $\mathbf{Q} = \mathbf{N}\mathbf{M}^{-1}\mathbf{N}^\mathsf{T}$ can be partitioned as follows:

$$\begin{bmatrix} \mathbf{Q}_{II} & \mathbf{Q}_{ID} \\ \mathbf{Q}_{DI} & \mathbf{Q}_{DD} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_I \\ \boldsymbol{z}_D \end{bmatrix} + \begin{bmatrix} \boldsymbol{q}_I \\ \boldsymbol{q}_D \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_I \\ \boldsymbol{w}_D \end{bmatrix} \tag{162}$$

Given some matrix $\boldsymbol{\gamma} \in \mathbb{R}^{(n-r)\times r}$, it is the case that $\mathbf{N}_D = \boldsymbol{\gamma}\mathbf{N}_I$, and therefore that $\mathbf{Q}_{DI} = \boldsymbol{\gamma}\mathbf{N}_I\mathbf{M}^{-1}{\mathbf{N}_I}^\mathsf{T}$, $\mathbf{Q}_{ID} = \mathbf{N}_I\mathbf{M}^{-1}{\mathbf{N}_I}^\mathsf{T}\boldsymbol{\gamma}^\mathsf{T}$ (by symmetry),
$\mathbf{Q}_{DD} = \boldsymbol{\gamma}\mathbf{N}_I\mathbf{M}^{-1}{\mathbf{N}_I}^\mathsf{T}\boldsymbol{\gamma}^\mathsf{T}$, and $\boldsymbol{q}_D = \boldsymbol{\gamma}\mathbf{N}_I\boldsymbol{v}$.

Lemma 10.2 *Since rank*$(\mathbf{NM}) \leq \min(\textit{rank}(\mathbf{N}), \textit{rank}(\mathbf{M}))$*, the number of positive components of* $\boldsymbol{z}_I$ *can not be greater than rank*$(\mathbf{N})$.

Proof The columns of $\mathbf{NM}$ have $\mathbf{N}$ multiplied by each column of $\mathbf{M}$, *i.e.*, $\mathbf{NM} = \begin{bmatrix} \mathbf{N}\boldsymbol{m}_1 & \mathbf{N}\boldsymbol{m}_2 & \dots & \mathbf{N}\boldsymbol{m}_m \end{bmatrix}$. Columns in $\mathbf{M}$ that are linearly dependent will thus produce columns in $\mathbf{NM}$ that are linearly dependent (with precisely the same coefficients). Thus, rank($\mathbf{NM}$) $\leq$ rank($\mathbf{M}$). Applying the same argument to the transposes produces rank($\mathbf{NM}$) $\leq$ rank($\mathbf{N}$), thereby proving the claim. ∎

The following shows that no more positive force magnitudes are necessary to solve the LCP in the case that the number of positive components of $\boldsymbol{z}_I$ is equal to the rank of $\mathbf{N}$.

Theorem 10.3 *If* $(\boldsymbol{z}_I = \boldsymbol{a}, \boldsymbol{w}_I = \mathbf{0})$ *is a solution to the LCP* $(\boldsymbol{q}_I, \mathbf{Q}_{II})$*, then* $(\begin{bmatrix} {\boldsymbol{z}_I}^\mathsf{T} = \boldsymbol{a}^\mathsf{T} & {\boldsymbol{z}_D}^\mathsf{T} = \mathbf{0}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \boldsymbol{w} = \mathbf{0})$ *is a solution to the LCP* $(\boldsymbol{q}, \mathbf{Q})$.

Proof For $(\begin{bmatrix} {\boldsymbol{z}_I}^\mathsf{T} = \boldsymbol{a}^\mathsf{T} & {\boldsymbol{z}_D}^\mathsf{T} = \mathbf{0}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \boldsymbol{w} = \mathbf{0})$ to be a solution to the LCP $(\boldsymbol{q}, \mathbf{Q})$, six conditions must be satisfied:

- $\boldsymbol{z}_I \geq \mathbf{0}$
- $\boldsymbol{w}_I \geq \mathbf{0}$

- $\boldsymbol{z}_I{}^{\mathsf{T}}\boldsymbol{w}_I = 0$
- $\boldsymbol{z}_D \geq \mathbf{0}$
- $\boldsymbol{w}_D \geq \mathbf{0}$
- $\boldsymbol{z}_D{}^{\mathsf{T}}\boldsymbol{w}_D = 0$

Of these, (1), (4), and (6) are met trivially by the assumptions of the theorem. Since $\boldsymbol{z}_D = \mathbf{0}$, $\mathbf{Q}_{II}\boldsymbol{z}_I + \mathbf{Q}_{ID}\boldsymbol{z}_D + \boldsymbol{q}_I = \mathbf{0}$, and thus $\boldsymbol{w}_I = \mathbf{0}$, thus satisfying (2) and (3). Also due to $\boldsymbol{z}_D = \mathbf{0}$, it suffices to show for (5) that $\mathbf{Q}_{DI}\boldsymbol{z}_I + \boldsymbol{q}_D \geq \mathbf{0}$. From above, the left hand side of this equation is equivalent to $\boldsymbol{\gamma}(\mathbf{N}_I\mathbf{M}^{-1}\mathbf{N}_I{}^{\mathsf{T}}\boldsymbol{a} + \mathbf{N}_I\boldsymbol{v})$, or $\boldsymbol{\gamma}\boldsymbol{w}_I$, which itself is equivalent to $\boldsymbol{\gamma}\mathbf{0}$. Thus, $\boldsymbol{w}_D = \mathbf{0}$.

Appendix E: Scoring each inverse dynamics controller implementation

Task	No-slip (no impact) **ID**$(t_i)_{LCP,\infty}$ (Section 7.3)	No-slip (impact model) **ID**$(t_i)_{QP,\infty}$ (Section 7.5)	Coulomb friction (no impact) **ID**$(t_i)_{LCP,\mu}$ (Section 7.4)	Coulomb friction (impact model) **ID**$(t_i)_{QP,\mu}$ (Section 7.5)
Biped walking / quadruped two-footed gait high friction terrain $[0-2]$ **contacts**	**PASS** † + continuous contact forces + distributed contact forces + fastest computation	**PASS** + continuous contact forces + distributed contact forces − slower computation	**PASS** + continuous contact forces + distributed contact forces − slower computation	**PASS** + continuous contact forces + distributed contact forces − slowest computation
Biped walking / quadruped two-footed gait on low friction terrain $[0-2]$ **contacts**	**FAIL** (slip and fall) − excessive contact forces	**FAIL** (slip and fall) − excessive contact forces	**PASS** † + continuous contact forces + distributed contact forces − slower computation	**PASS** + continuous contact forces + distributed contact forces − slowest computation
Quadruped (or greater number of legs) walking on high friction terrain $[2-n]$ **contacts**	**PASS** + continuous contact forces − undistributed contact forces + fastest computation	**PASS** † + continuous contact forces + distributed contact forces − slower computation	**FAIL** (torque chatter) − discontinuous contact forces − undistributed contact forces	**PASS** + continuous contact forces + distributed contact forces − slowest computation
Quadruped (or greater number of legs) walking on low friction terrain $[2-n]$ **contacts**	**FAIL** (slip and fall) − excessive contact forces	**FAIL** (slip and fall) − excessive contact forces	**FAIL** (torque chatter) − discontinuous contact forces − undistributed contact forces	**PASS** † + continuous contact forces + distributed contact forces
Fixed base manipulator grasping a high friction object $[2-n]$ **contacts**	**PASS** + continuous contact forces − undistributed contact forces + fastest computation	**PASS** † + continuous contact forces + distributed contact forces − slower computation	**FAIL** (torque chatter) − discontinuous contact forces − undistributed contact forces	**PASS** + continuous contact forces + distributed contact forces − slowest computation
Fixed base manipulator grasping a low friction object $[2-n]$ **contacts**	**FAIL** (slip and fall) − excessive contact forces	**FAIL** (slip and fall) − excessive contact forces	**FAIL** (torque chatter) − discontinuous contact forces − undistributed contact forces	**PASS** † + continuous contact forces + distributed contact forces

Table 18: *A table describing the behavior of each inverse dynamics controller implementation when used to control disparate robot morphologies through different tasks. If the robot performed the task without failing any of the performance criteria (no torque chatter, no falling) it is marked as a pass; Otherwise, the task will be marked as a failure for the reason noted in parenthesis.*
†: Indicates which inverse dynamics implementation that were determined to be the best controller for the example task, prioritizing: (1) [critical] Successful performance of the task; (2) [critical] Mitigation of torque chatter (continuous contact forces); (3) [non-critical] Even distribution of contact forces (distributed contact forces); (4) [non-critical] Computation speed.

References

H. Abbas, et al. (2013). 'Probabilistic Temporal Logic Falsification of Cyber-Physical Systems'. *ACM Transactions on Embedded Computing Systems* **12**(2).

A. Ames (2013). 'Human-Inspired Control of Bipedal Robotics via Control Lyapunov Functions and Quadratic Programs'. In *Proc. Intl. Conf. Hybrid Systems: Computation and Control*.

M. Anitescu (2006). 'Optimization-based simulation of nonsmooth dynamics'. *Mathematical Programming, Series A* **105**:113–143.

M. Anitescu & G. D. Hart (2004). 'A Constraint-Stabilized Time-Stepping Approach for Rigid Multibody Dynamics with Joints, Contacts, and Friction'. *Intl. Journal for Numerical Methods in Engineering* **60**(14):2335–2371.

M. Anitescu & F. A. Potra (1997). 'Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems'. *Nonlinear Dynamics* **14**:231–247.

M. Anitescu & F. A. Potra (2002). 'A Time-Stepping Method for Stiff Multi-Rigid-Body Dynamics with Contact and Friction'. *Intl. Journal for Numerical Methods in Engineering* **55**:753–784.

U. M. Ascher, et al. (1995). 'Stabilization of constrained mechanical systems with DAEs and invariant manifolds'. *J. Mech. Struct. Machines* **23**:135–158.

J. E. Auerbach & J. C. Bongard (2012). 'On the Relationship Between Environmental and Mechanical Complexity in Evolved Robots'. In *Intl. Conf. on the Synthesis and Simulation of Living Systems (ALife)*, vol. 13, pp. 309–316.

D. Aukes & M. R. Cutkosky (2013). 'Simulation-Based Tools for Evaluating Underactuated Hand Designs'. In *Proc. IEEE Intl. Conf. Robotics Automation (ICRA)*, Karlsruhe, Germany.

D. M. Aukes, et al. (2014). 'Design and testing of a selectively compliant underactuated hand'. *Intl. J. Robot. Res.* **33**(5).

D. Baraff (1994). 'Fast Contact Force Computation for Nonpenetrating Rigid Bodies'. In *Proc. of SIGGRAPH*, Orlando, FL.

V. Barasuol, et al. (2013). 'A reactive controller framework for quadrupedal locomotion on challenging terrain'. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*, Karlsruhe, Germany.

R. Barzel, et al. (1996). 'Plausible Motion Simulation for Computer Graphics Animation'. In R. Boulic & G. Hégron (eds.), *Computer Animation and Simulation (Proc. Eurographics Workshop)*, pp. 183–197.

J. Baumgarte (1972). 'Stabilization of constraints and integrals of motion in dynamical systems'. *Comp. Math. Appl. Mech. Engr.* **1**:1–16.

A. Bemporad & M. Morari (2007). *Robust model predictive control: A survey*, vol. 245 of *Lecture Notes in Control and Information Sciences*, chap. Robust model predictive control: A survey, pp. 207–226. Springer London, London.

R. Bhatia (2007). *Positive definite matrices*. Princeton University Press.

R. W. Bisseling & A. L. Hof (2006). 'Handling of impact forces in inverse dynamics'. *J. Biomech.* **39**(13):2438–2444.

L. Blackmore (2006). 'A probabilistic particle control approach to optimal, robust predictive control'. In *In Proceedings of the AIAA Guidance, Navigation and Control Conference*.

W. Blajer, et al. (2007). 'Multibody modeling of human body for the inverse dynamics analysis of sagittal plane movements'. *Multibody System Dynamics* **18**(2):217–232.

J. C. Bongard (2014). 'Why Morphology Matters'. *The Horizons of Evolutionary Robotics* pp. 125–152.

S. Boyd & L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

M. S. Branicky, et al. (2006). 'Sampling-based planning, control and verification of hybrid systems'. In *IEE Proceedings - Control Theory and Applications*.

B. Brogliato (1996). *Nonsmooth Impact Mechanics: Models, Dynamics, and Control*. Springer-Verlag, London.

C. J. Budd (1996). 'Non-smooth dynamical systems and the grazing bifurcation'. In *Nonlinear Mathematics and its Applications*, pp. 219–235. Cambridge Univ. Press.

S. A. Burden, et al. (2015). 'Near–Simultaneous Footfalls Lend Stability to Multi–Legged Gaits'. In *Dynamic Walking*.

A. Cangelosi, et al. (2015). *Handbook of Computational Intelligence*, chap. Embodied intelligence, pp. 697–714. Springer.

A. Chatterjee (1999). 'On the realism of complementarity conditions in rigid-body collisions'. *Nonlinear Dynamics* **20**:159–168.

A. Chatterjee & A. Ruina (1998). 'A New Algebraic Rigid Body Collision Law Based on Impulse Space Considerations'. *ASME J. Appl. Mech.* **65**(4):939–951.

N. Cheney, et al. (2016). 'On the difficulty of co-optimizing morphology and control in evolved virtual creatures'. In *Intl. Conf. on the Synthesis and Simulation of Living Systems (ALife)*, vol. 15, Cancun, Mexico.

S. H. Collins, et al. (2001). 'A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees'. *Intl. J. Robot. Res.* **20**(2).

S. Coros, et al. (2011). 'Locomotion Skills for Simulated Quadrupeds'. In *Proc. ACM SIGGRAPH*.

R. W. Cottle (1968). 'The Principal Pivoting Method of Quadratic Programming'. In G. Dantzig & J. A. F. Veinott (eds.), *Mathematics of Decision Sciences*, pp. 144–162. AMS, Rhode Island.

R. W. Cottle, et al. (1992). *The Linear Complementarity Problem*. Academic Press, Boston.

N. B. Do, et al. (2007). 'Efficient Simulation of a Dynamic System with LuGre Friction'. *J. of Computational and Nonlinear Dynamics* **2**:281–289.

M. Dorigo & M. Colombetti (1994). 'Robot shaping: Developing situated agents through learning'. *Artificial Intelligence* **70**(2):321–370.

E. Drumwright & D. A. Shell (2010). 'Modeling Contact Friction and Joint Friction in Dynamic Robotic Simulation using the Principle of Maximum Dissipation'. In *Proc. of Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

D. J. Duff, et al. (2011). 'Physical simulation for monocular 3D model based tracking'. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*.

C. Ericson (2005). *Real-Time Collision Detection*. Morgan Kaufmann.

J. M. Esposito, et al. (2005). *Algorithmic Foundations of Robotics VI*, vol. 17 of *Springer Tracts in Advanced Robotics*, chap. Adaptive RRTs for Validating Hybrid Robotic Control Systems, pp. 107–121. Springer.

P. L. Fackler & M. J. Miranda (2011). 'LEMKE'. http://www4.ncsu.edu/ pfackler/compecon/toolbox.html.

R. Featherstone (1987). *Robot Dynamics Algorithms*. Kluwer.

R. Featherstone (2004). 'An Empirical Study of the Joint Space Inertia Matrix'. *The Intl. J. of Robotics Research* **23**(9):859–871.

R. Featherstone (2008). *Rigid Body Dynamics Algorithms*. Springer.

S. Feng, et al. (2013). '3D Walking Based on Online Optimization'. In *Proc. IEEE-RAS Intl. Conf. on Humanoid Robots (HUMANOIDS)*, Atlanta, GA.

C. Gehring, et al. (2013). 'Control of dynamic gaits for a quadrupedal robot'. In *2013 IEEE International Conference on Robotics and Automation*, pp. 3287–3292.

E. Hairer & G. Wanner (1996). *Solving ordinary differential equations II: stiff and differential-algebraic problems, 2nd. ed*. Springer Verlag, Berlin.

H. Hatze (2002). 'The fundamental problem of myoskeletal inverse dynamics and its implications'. *J. Biomech.* **35**(1):109–115.

B. Hengst, et al. (2002). 'Omnidirectional Locomotion for Quadupred Robots'. In *Proc. RoboCup 2001: Robot Soccer World Cup V*, pp. 368–373.

H. M. Herr, et al. (2002). 'A model of scale effects in mammalian quadrupedal running'. *J. Experimental Biology* **205**:959–967.

R. Hunger (2007). 'Floating Point Operations in Matrix-Vector Calculus'. Tech. rep., TU München.

M. Hutter & R. Siegwart (2012). 'Hybrid Operational Space Control for Compliant Quadruped Robots'. In *Proc. Dynamic Walking*.

M. Hutter, et al. (2014). 'Quadrupedal locomotion using hierarchical operational space control'. *Intl. J. Robot. Res.* **33**(8):1047–1062.

A. P. Ivanov (1995). 'On multiple impact'. *J. Applied Mathematics and Mechanics* **59**(6):887–902.

A. M. Johnson, et al. (2016). 'Convergent Planning'. *IEEE Robotics and Automation Letters* **1**(2):1044–1051.

B. Johnson & H. Kress-Gazit (2015). 'Analyzing and revising synthesized controllers for robots with sensing and actuation errors'. *Intl. J. Robot. Res.* **34**:816–832.

M. Kalakrishnan, et al. (2011a). 'Learning, Planning, and Control for Quadruped Locomotion over Challenging Terrain'. *Intl. J. Robot. Res.* **30**(2):236–258.

M. Kalakrishnan, et al. (2011b). 'STOMP: Stochastic trajectory optimization for motion planning'. In *2011 IEEE International Conference on Robotics and Automation*, pp. 4569–4574.

T. R. Kane & D. A. Levinson (1985). *Dynamics: Theory and Applications*. McGraw-Hill, New York.

G. Kewlani, et al. (2009). 'Stochastic mobility-based path planning in uncertain environments'. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1183–1189. IEEE.

J. Kim, et al. (2013). 'Physically Based Grasp Quality Evaluation under Pose Uncertainty'. *IEEE TRANSACTIONS ON ROBOTICS* **29**(6).

N. Koenig & A. Howard (2004). 'Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator'. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2149–2154, Sendai, Japan.

T. Koolen, et al. (2012). 'Capturability-based Analysis and Control of Legged Locomotion, Part 1: Theory and Application to Three Simple Gait Models'. *Int. J. Rob. Res.* **31**(9):1094–1113.

S. Koos, et al. (2013a). 'Fast Damage Recovery in Robotics with the T-resilience Algorithm'. *Int. J. Rob. Res.* **32**(14):1700–1723.

S. Koos, et al. (2013b). 'The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics'. *IEEE Transactions on Evolutionary Computation* **17**(1):122–145.

M. Koval, et al. (2013). 'Pose Estimation for Contact Manipulation with Manifold Particle Filters'. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*.

S. Kuindersma, et al. (2014). 'An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion'. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*.

A. D. Kuo (1998). 'A least-squares estimation approach to improving the precision of inverse dynamics calculations'. *Trans. American Society Mech. Engineers (ASME) J. Biomech. Engr.* **120**:148–159.

C. Lacoursière (2003). 'Splitting Methods for Dry Frictional Contact Problems in Rigid Multibody Systems: Preliminary Performance Results'. In M. Ollila (ed.), *Proc. of SIGRAD*, pp. 11–16.

C. Lacoursière (2007). *Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts*. Ph.D. thesis, Umeå University.

S. LaValle & J. J. Kuffner, Jr. (2001). 'Randomized Kinodynamic Planning'. *Intl. J. of Robotics Research* **20**(5):378–400.

R. Leine & N. van de Wouw (2008a). *Stability and convergence of mechanical systems with unilateral constraints*. Springer Verlag.

R. Leine & N. van de Wouw (2008b). 'Stability properties of equilibrium sets of non-linear mechanical systems with dry friction and impact'. *Nonlinear Dynamics* **51**(4):551–583.

R. I. Leine & C. Glocker (2003). 'A set-valued force law for spatial Coulomb-Contensou friction'. *Europen J. of Mechanics/A Solids* **22**(2):193–216.

C. E. Lemke (1965). 'Bimatrix Equilibrium Points and Mathematical Programming'. *Management Science* **11**:681–689.

S. Li, et al. (2015a). 'State Estimation for Dynamic Systems with Intermittent Contact'. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*.

S. Li, et al. (2015b). 'A Comparative Study of Contact Models for Contact-Aware State Estimation'. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*.

W. Lohmiller & J.-J. E. Slotine (1998). 'On Contraction Analysis for Non-linear Systems'. *Automatica* **34**(6):683–696.

K. Lynch & M. J. Mason (1995). 'Pushing by slipping, Slip with infinite friction, and perfectly rough surfaces'. *Intl. J. Robot. Res.* **14**(2):174–183.

J. Mahler, et al. (2015). 'GP-GPIS-OPT: Grasp Planning With Shape Uncertainty Using Gaussian Process Implicit Surfaces and Sequential Convex Programming'. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

O. L. Mangasarian & S. Fromovitz (1967). 'The Fritz John Necessary Optimality Conditions in the Presence of Equality and Inequality Constraints'. *J. Mathematical Analysis and Appl.* **17**:37–47.

D. L. Marruedo, et al. (2002). 'Input-to-state stable MPC for constrained discrete-time nonlinear systems with bounded additive uncertainties'. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 4, pp. 4619–4624 vol.4.

N. A. Melchior & R. Simmons (2007). 'Particle RRT for path planning with uncertainty'. In *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1617–1624. IEEE.

B. Mirtich (1996). *Impulse-based Dynamic Simulation of Rigid Body Systems*. Ph.D. thesis, University of California, Berkeley.

M. Mistry, et al. (2010). 'Inverse Dynamics Control of Floating Base Systems Using Orthogonal Decomposition'. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3406–3412.

K. Mombaur, et al. (2005). 'Open-loop stable running'. *Robotica* **23**(1).

I. Mordatch, et al. (2015). 'Ensemble-CIO: Full-Body Dynamic Motion Planning that Transfers to Physical Humanoids'. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

I. Mordatch, et al. (2012). 'Discovery of complex behaviors through contact-invariant optimization.'. *ACM Trans. Graph.* **31**(4):43.

I. Mordatch, et al. (2013). 'Animating Human Lower Limbs Using Contact-Invariant Optimization'. *ACM Trans. on Graphics* **32**(6).

J. J. Moreau (1983). *Standard inelastic shocks and the dynamics of unilateral constraints*, pp. 173–221. Springer-Verlag, New York.

K. G. Murty (1988). *Linear Complementarity, Linear and Nonlinear Programming*. Heldermann Verlag, Berlin.

A. L. Nelson, et al. (2009). 'Fitness functions in evolutionary robotics: A survey and analysis'. *Robotics and Autonomous Systems* **57**(4):345—370.

P. E. Nikravesh (1988). *Computer-Aided Analysis of Mechanical Systems*. Prentice Hall.

J. Nocedal & S. J. Wright (2006). *Numerical Optimization, 2nd ed.* Springer-Verlag.

C. Ott, et al. (2011). 'Posture and Balance Control for Biped Robots based on Contact Force Optimization'. In *Proc. IEEE-RAS Intl. Conf. on Humanoid Robots (HUMANOIDS)*.

P. Painlevé (1895). 'Sur le lois du frottement de glissemment'. *C. R. Académie des Sciences Paris* **121**:112–115.

J.-S. Pang & D. E. Stewart (2008). 'Differential variational inequalities'. *Math. Program., Ser. A* **113**:345–424.

A. Papachristodoulou & S. Prajna (2009). 'Robust Stability Analysis of Nonlinear Hybrid Systems'. *IEEE Trans. Autom. Control* **54**(5).

S. Patil, et al. (2014). 'Gaussian belief space planning with discontinuities in sensing domains'. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6483–6490. IEEE.

J. Peters, et al. (2008). 'A unifying methodology for robot control with redundant DOFs'. *Autonomous Robots* **24**(1–12).

R. Pfeifer & F. Iida (2009). *Creating Brain-Like Intelligence*, vol. 5436 of *Lecture Notes in Computer Science*, chap. Morphological Computation: Connecting Body, Brain, and Environmentand environment, pp. 66–83. Springer.

R. Platt Jr, et al. (2010). 'Belief space planning assuming maximum likelihood observations' .

B. Ponton, et al. (2016). 'Risk sensitive nonlinear optimal control with measurement uncertainty'. *CoRR* **abs/1605.04344**.

M. Posa, et al. (2014). 'A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact'. *Int. J. Rob. Res.* **33**(1):69–81.

M. Posa & R. Tedrake (2012). 'Direct Trajectory Optimization of Rigid Body Dynamical Systems through Contact'. In *Proc. Workshop on Algorithmic Foundations of Robotics (WAFR)*, Boston.

M. Posa, et al. (2015). 'Stability Analysis and Control of Rigid Body Systems with Impacts and Friction'. *IEEE Trans. Autom. Control* .

S. Prajna, et al. (2007). 'A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates'. *IEEE Trans. Autom. Control* **52**(8).

S. Prajna & A. Rantzer (2007). 'Convex Programs for Temporal Verification of Nonlinear Dynamical Systems'. *SIAM J. Control Optim.* .

S. Prentice & N. Roy (2009). 'The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance'. *The International Journal of Robotics Research* **28**(11-12):1448–1465.

W. H. Press, et al. (1992). *Numerical Recipes in C*. Cambridge University Press, second edn.

F. Qian & D. Goldman (2015). 'Scattering of a legged robot in a heterogeneous granular terrain'. *Bulletin of the American Physical Society* **60**.

M. H. Raibert (1986). *Legged Robots That Balance*. MIT Press, Cambridge, MA.

L. Righetti, et al. (2013). 'Optimal distribution of contact forces with inverse-dynamics control'. *Intl. J. Robot. Res.* **32**(3):280–298.

L. Righetti, et al. (2011). 'Inverse Dynamics Control of Floating-Base Robots with External Constraints: A Unified View'. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China.

E. Rombokas, et al. (2012). 'Biologically inspired grasp planning using only orthogonal approach angles'. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 1656–1661.

L. Saab, et al. (2013). 'Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints'. *IEEE Trans. Robotics* **29**(2).

O. Saglam & K. Byl (2014). 'Robust Policies via Meshing for Metastable Rough Terrain Walking'. In *Robotics: Science and Systems (RSS)*.

R. W. H. Sargent (1978). 'An efficient implementation of the Lemke Algorithm and its extension to deal with upper and lower bounds'. *Mathematical Programming Study* **7**:36–54.

B. Satzinger, et al. (2014). 'More Solutions Means More Problems: Resolving Kinematic Redundancy in Robot Locomotion on Complex Terrain'. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*, Chicago.

A. B. Sawers & M. E. Hahn (2010). 'The Potential for Error with Use of Inverse Dynamic Calculations in Gait Analysis of Individuals with Lower Limb Loss: A Review of Model Selection and Assumptions'. *J. Prosthetics & Orthotics* **22**(1):56–61.

S. Schaal (2009). 'The SL simulation and real-time control software package'. Tech. rep., Univ. Southern California.

A. T. Schwarm & M. Nikolaou (1999). 'Chance-constrained model predictive control'. *AIChE Journal* **45**(8):1743–1752.

L. Sciavicco & B. Siciliano (2000). *Modeling and Control of Robot Manipulators, 2nd Ed.* Springer-Verlag, London.

J. Shen & J. S. Pang (2005). 'Linear Complementarity Systems: Zeno States'. *SIAM J. on Control and Optimization* **44**(3):1040–1066.

K. Sims (1994). 'Evolving Virtual Creatures'. In *Special Interest Group on Computer GRAPHics and Interactive Techniques (SIGGRAPH)*, pp. 15–22.

B. Smith, et al. (2012). 'Reflections on Simultaneous Impact'. *ACM Trans. on Graphics (Proc. of SIGGRAPH)* **31**(4):106:1–106:12.

J. Smith, et al. (2014). 'Momentum-based whole body control framework—application to the humanoid robots Atlas and Valkyrie'. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Systems (IROS) workshop on Whole-Body Control for Robots in the Real World*.

B. Stephens & C. Atkeson (2010a). 'Dynamic Balance Force Control for Compliant Humanoid Robots'. In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS)*.

B. J. Stephens & C. G. Atkeson (2010b). 'Push Recovery by stepping for humanoid robots with force controlled joints'. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pp. 52–59.

D. Stewart & J. C. Trinkle (2000). 'An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction'. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, San Francisco, CA.

D. E. Stewart (1998). 'Convergence of a time-stepping scheme for rigid-body dynamics and resolution of Painlevé's problem'. *Arch. Ration. Mech. Anal.* **145**:215–260.

D. E. Stewart (2000a). 'Rigid-body Dynamics with Friction and Impact'. *SIAM Review* **42**(1):3–39.

D. E. Stewart (2000b). 'Time-stepping methods and the mathematics of rigid body dynamics'. In A. Guran, B. Feeny, A. Klarbring, & Y. Ishida (eds.), *Impact and Friction of Solids, Structures, and Intelligent Machines*. World Scientific.

D. E. Stewart & J. C. Trinkle (1996). 'An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction'. *Intl. Journal for Numerical Methods in Engineering* **39**(15):2673–2691.

R. Stribeck (1902). 'Die wesentlichen Eigenschaften der Gleit und Rollenlager (The key qualities of sliding and roller bearings)'. *Zeitschrift des Vereines Seutscher Ingenieure* **46**(38–39):1342–1348.

T. Sugihara & Y. Nakamura (2003). 'Variable Impedant Inverted Pendulum Model Control for a Seamless Contact Phase Transition on Humanoid Robot'. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*.

J. R. Taylor & E. M. Drumwright (2016). 'State Estimation of a Wild Robot Toward Validation of Rigid Body Simulation'. In *Proc. Intl. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, San Francisco, CA.

J. R. Taylor, et al. (2014). 'Making Time Make Sense in Robotic Simulation'. In *Proc. Intl. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Bergamo, Italy.

R. Tedrake & the Drake Development Team (2016). 'Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems'.

E. Todorov (2011). 'A convex, smooth and invertible contact model for trajectory optimization'. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai.

E. Todorov (2014). 'Analytically-invertible dynamics with contacts and constraints: theory and implementation in MuJoCo'. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*.

C. J. Tomlin, et al. (2000). 'A Game Theoretic Approach to Controller Design for Hybrid Systems'. *Proc. IEEE* **88**:949–969.

C. J. Tomlin, et al. (2003). 'Computational Techniques for the Verification of Hybrid Systems'. *Proc. of the IEEE* **91**(7).

J. Trinkle, et al. (1997). 'On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction'. *Zeithscrift fur Angewandte Mathematik und Mechanik* **77**(4):267–279.

C. D. Twigg & D. L. James (2007). 'Many-worlds Browsing for Control of Multibody Dynamics'. *ACM Trans. on Graphics* **26**(3).

J. van den Berg, et al. (2011). 'LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information'. *The International Journal of Robotics Research* **30**(7):895–913.

A. J. Van Den Bogert & A. Su (2008). 'A weighted least squares method for inverse dynamic analysis'. *Comput. Methods Appl. Mech. Eng.* **11**(1):3–9.

V. Venkatasubramanian, et al. (1993). 'Analysis of local bifurcation mechanisms in large differential-algebraic systems such as the power system'. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pp. 3727–3733 vol.4.

R. Walters, et al. (2002). *Uncertainty Analysis for Fluid Mechanics with Applications*. ICASE report. ICASE, NASA Langley Research Center.

J. Wang, et al. (2009). 'Robot Jenga: Autonomous and Strategic Block Extraction'. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO.

J. Weisz & P. K. Allen (2012). 'Pose Error Robust Grasping from Contact Wrench Space Metrics'. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

D. Xiu & J. S. Hesthaven (2005). 'High-Order Collocation Methods for Differential Equations with Random Inputs'. *SIAM Journal on Scientific Computing* **27**(3):1118–1139.

J. Yang, et al. (2007). 'An Inverse Dynamical Model for Slip Gait'. In *Proc. First Intl. Conf. Digital Human Modeling*, Beijing, China.

K.-T. Yu, et al. (2016). 'More than a million ways to be pushed. A high-fidelity experimental dataset of planar pushing'. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 30–37. IEEE.

S. Zapolsky (2015). 'Pacer'. https://github.com/PositronicsLab/Pacer.

S. Zapolsky & E. Drumwright (2014). 'Quadratic Programming-based Inverse Dynamics Control for Legged Robots with Sticking and Slipping Frictional Contacts'. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*.

S. Zapolsky, et al. (2013). 'Inverse Dynamics for a Quadruped Robot Locomoting on Slippery Surfaces'. In *Proc. Intl. Conf. Climbing Walking Robots (CLAWAR)*, Sydney, Australia.

S. Zapolsky & E. M. Drumwright (2015). 'Adaptive Integration for Controlling Speed vs. Accuracy in Multi-Rigid Body Simulation'. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*.

L. E. Zhang, et al. (2013). 'A Dynamic Bayesian Approach to Real-Time Estimation and Filtering in Grasp Acquisition'. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*.

H. Zhao, et al. (2014). 'Human-inspired multi-contact locomotion with amber2'. In *ACM/IEEE, International Conference on Cyber Physics System*.

Y. Zheng & W.-H. Qian (2005). 'Coping with the Grasping Uncertainties in Force-closure Analysis'. *Int. J. Rob. Res.* **24**(4):311–327.

S. Zickler & M. Veloso (2009). 'Efficient Physics-Based Planning: Sampling Search Via Non-Deterministic Tactics and Skills'. In *Proc. Autonomous Agents and Multiagent Systems (AAMAS)*.