Unified Foothold Selection and Motion Planning for Legged Systems in Real-Time over Rough Terrain

Submitted in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Department of Mechanical Engineering

Steven R. Crews II

B.S., Mechanical Engineering, United States Military Academy
M.S., Astronautical Engineering, Naval Postgraduate School
M.S., Mechanical Engineering, Naval Postgraduate School
Astronautical Engineer, Naval Postgraduate School

Carnegie Mellon University
Pittsburgh, PA

May, 2020

To my wife and my kids.

Acknowledgements

I want to thank Howie Choset and Matt Travers for bringing me into their Biorobotics Lab and introducing me to the world of robotics. The exposure to so many problems that the members of the lab tackle really opened my eyes as to what robotics is all about. After a year, I completely forgot who was an EE, MechE, RI, or CS. It was amazing to see everyone learn enough of all of these fields to be effective roboticists. You two set an astounding example of what a leadership team can be. Howie, thank you for all of the opportunities to demonstrate robotic capabilities to VIPs, the advice for my future as a professor, and serving on my doctoral committee. Matt, thank you for serving as my doctoral committee chair and thesis advisor. You guided me through this process for the past few years and opened my eyes as to the breadth of our field and the endless possibilities for implementation.

Mark Bedillion and Koushil Sreenath, thank you for serving as members on my doctoral committee. Mark, it was an honor serving as your teaching assistant and your advice as to my future as a professor was well-received. Koushil, you got me started here at CMU and your expertise in legged systems has helped me truly understand how difficult dynamic implementation of higher order systems can be, especially on actual hardware!

Thank you to the students, staff, and faculty that I worked so closely with over the years. Sapan Agrawal, your collaboration was instrumental to getting this method developed and derived. Yusuf Senturk, our entire summer of study sessions absolutely got me through the Ph.D. controls qualifications. Adhitya Ganesh, Abhimanyu Singh and Zixin Zhang, the robot wouldn't be walking without the countless hours you spent in the early days of the hardware gait design. Daniel Piedrahita and Joey Wood, thank you

for leaving me a solid robot to initialize my locomotion research. Joe Norby, considering a Minitaur implementation really helped me better understand how to implement this method. Cosette Craig and Kyuto Furutachi, thanks for the assistance over the summer. Small projects pay big dividends down the road. Naman Gupta and Jackie Grover, all of those late nights pouring through DDP derivations finally paid off. Evan Harber, your foot sensors changed the game, and your assistance with ROS integration finally taught me this wonderful tool. Julian, thanks for your time and help with whatever problem I had. As a fellow Ph.D. Candidate, you helped me far more than I helped you. Mickey Velado and Nate Shoemaker, we almost got the upgraded Flyped flywheel built, almost. It still is an awesome idea and I hope you get it up and running. Lu Li, Jim Picard and Charlie Hart, thanks so much for your precious time and invaluable advice. Aaron Johnson, your Robot Design and Experimentation course was an incredible hands-on experience with designing a robot from scratch. Someday, I hope to continue with the falling cat robot.

I want to thank the U.S. Army for giving me this opportunity to attend CMU. You paid for me to be here and asked nothing of me while I attended school. As to the wealth of knowledge that CMU has imparted upon me, I hope that I can bring enough of it back to help my fellow Soldiers come home safely.

And finally, most of all, I want to thank my family. To Tara, my wife. You have been amazing and supportive over the years, especially these last few. You made my school our family's priority, which is a lot to ask of a spouse. I wouldn't have made it through without your love and care. Thank you. To my kids, Abby, Carlie and Gavin. Your daddy has been in perpetual school alongside you. Hopefully, I set a good student example for you. We've missed a lot of time together while I was doing research and building robots. I promise to make it up to you. I love you all so much.

Abstract

The greatest potential for humanoid robots is that they will someday be able to robustly traverse and perform meaningful work in a world designed by and for human beings. Thus, humanoids, and legged platforms more generally, offer the practical potential to bring elements of automation off the factory floor and aspects of artificial intelligence into the physical world. However, in practice, legged systems remain quite brittle when operating in unstructured terrains. The primary issue continues to be that it is difficult to carefully deliberate about how to make progress towards a goal while maintaining balance over the time scales necessary to remain dynamically stable. Therefore, this work presents a novel framework for planning and controlling in closed-loop the behavior of legged robots moving fluidly through unstructured terrains. Specifically, the framework we put forth unifies the commonly disparate components of footstep planning, motion planning, and feedback control that we show can be used to robustly plan the motions of humanoid robots moving through a variety of complex terrains.

Planning and controlling the motion of humanoid robots is complicated by the fact that they make and break contact with the environment. These discrete interruptions to the continuous dynamics mean that legged systems have hybrid dynamics. This presents an issue because conventional techniques for robotic motion planning and control are designed to work for systems with continuous dynamics. Therefore, in this work we establish a way to explicitly handle hybrid transitions within a trajectory optimization framework that has conventionally only been applied to continuous systems. More specifically, we solve a nonlinear trajectory optimization problem, wherein we assume a quadratic cost with nonlinear constraints imposed by the system's dynamics. To solve this problem,

we approximate the nonlinear constraints by computing the linearization of the system's hybrid dynamics. We show how to use tools from nonlinear analysis and control to compute analytically these approximations, despite the discontinuous nature of the trajectories about which the linearizations are computed. We follow a method very much inspired by existing approaches based on the sequential linear quadratic regulator to optimize over several hybrid steps at once. This allows us to reformulate planning footsteps and dynamically-feasible trajectories as well as deriving closed-loop controllers as a single trajectory optimization problem.

Posing our humanoid planning and control problem as a trajectory optimization formulation implicitly implies that we are solving for a locally optimal solution to the problem. Therefore, to ensure good resultant behavior and quick run time (reduce iterations to convergence), there must be a good way to introduce an initial stable nominal trajectory, or rather a *seed*, to the optimization. To this end, we create an intuitive means for generating such seeds that uses implicitly the "natural dynamics" of a given system. Our *energy management* technique ensures step-generation over newly-perceived terrain is dynamically feasible. The method additionally assures the system energy necessary to for future stepping.

Lastly, this work brings together both the unified planning and control approach and the energy-based seed generating algorithm to define the main contribution of this work: a unified planning architecture that is able to adapt to new complex, unstructured terrain online. As a legged system advances over new terrain, the energy method quickly determines footholds for newly-perceived terrain, seeding the unified planner for once-per-step online optimization. By introducing this energy management technique as the seed to our unified approach, we create a framework that allows a robot to safely adapt to rough terrain in real-time while respecting the natural dynamics of the system.

We present several walking models, each scaling up in complexity to demonstrate how this architecture can be applied to a family of different systems. In addition to analytical results and simulated demonstrations, we developed a hardware platform on which we validated our results.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

s.o.s. Surface of Section

CGW Compass Gait Walker

COM Center of Mass

CP Capture Point

DDP Differential Dynamic Programming

DH Denavit-Hartenberg

DOF Degree of Freedom

EKF Extended Kalman Filter

iLQR Iterative Linear Quadratic Regulator

KE Kinetic Energy

LIP Linear Inverted Pendulum

LQR Linear Quadratic Regulator

MPC Model Predictive Control

NIP Nonlinear Inverted Pendulum

PE Potential Energy

ROM Reduced Order Model

ROS Robot Operating System

SLAM Simultaneous Localization and Mapping

SLIP Spring Loaded Inverted Pendulum

SLQ Sequential Linear Quadratic

SOS Sum-of-Squares

ZMP Zero Moment Point

Symbols - Contiuous Space

$\boldsymbol{\Theta}$	Vector of Joint Angles ($\boldsymbol{\Theta} \in \mathbb{R}^{s \times 1}$)
$\boldsymbol{\tau}$	Control Torque Vector - Continuous Space
θ	Joint Angle
s	Number of Robot Joints
$\mathbf{q}$	State Vector - Continuous Space
ϕ	Event Function Marking End of Integration
f	Continuous Dynamics
N_τ	Number of Continuous Controls
N_q	Number of Continuous States

Symbols - Discrete Space

$\mathbf{x}$	State Vector - Once per Step Discrete Space
$\mathbf{u}$	Control Vector - Once per Step Discrete Space
$\mathbf{u}_I$	Discrete Phase Control Parameters - e.g. Once-per-Step Discrete Impulses
$\mathbf{u}_K$	Continuous Phase Control Parameters - e.g. Once-per-Step Change to Low Level Gains
N	Number of steps in planning horizon
S	Number of elements in a discretization of continuous space
F	State Transition Equation
M	Discrete Impact Mapping
P	Projection Mapping
D	Dimensionality Reduction Function
N_u	Number of Discrete Controls
N_x	Number of Discrete States

Symbols - Denavit-Hartenberg Convention

a_i	Length of i^{th} link
α_i	Twist of i^{th} link (not used in planar case)
d_i	Depth of i^{th} link (not used in planar case)
θ_i	Rotation at base of i^{th} link (right hand rotation convention)

m_i Mass of i^{th} link

l_{c_i} Length to center of mass of i^{th} link

j_i Moment of inertia of i^{th} link about the center of mass (scalar in planar case)

Symbols - Other

$\mathbb{R}$ Real Numbers

$\mathbb{Z}$ Integers

$\mathbf{y}$ Measurement vector

x translational position

z height

g Gravity

l Length

t Time

$\mathcal{L}$ Lagrangian

$\mathcal{T}$ Kinetic Energy

$\mathcal{U}$ Potential Energy

$\mathbf{Q}$ Configuration matrix

$\mathbf{K}$ Gain Matrix

Symbols - Cost Function

α Discount Factor

$\mathbf{Q}$ LQR control matrix with respect to states

$\mathbf{R}$ LQR control matrix with respect to control

$\mathcal{V}$ Value Function

Φ Terminal Cost

h Height Polynomial

J Cost Functional

p Quadratic Terrain Cost

L Running Cost

Superscripts

$+$ Moment After Impact

$-$ Moment Before Impact

* Star Refers to Nominal Point

$^{\mathrm{d}}$ desired or goal

$^{\mathrm{fb}}$ Feedback Increment

$^{\mathrm{ff}}$ Feedforward Increment

k Discrete Index

Subscripts

c Moment of Contact (i.e. time of contact t_c)

n Step Number

cp Capture Point

I Once per Step Impulses

K Once per Step Gain Selection

Other Symbols

$\bar{}$ Overbar refers difference from nominal (i.e. $\bar{\mathbf{q}} = \mathbf{q} - \mathbf{q}^*$)

$\dot{}$ Dot refers to rate of change with respect to time (i.e. $\dot{\mathbf{q}}$)

Chapter 1

Introduction

While traversing complex terrains, legged animals are constantly making decisions about how and where to place their feet while simultaneously maintaining their balance. To achieve human-like performance in complex terrains, this work supports the belief that walking robots need need to reason over a similarly complex decision making process to that employed by humans while actively maintaining their dynamic stability. To this end, this work develops a framework that enables legged robots to plan and then execute complex locomotive behaviors that can be rapidly adapted online to adjust for unexpected task or environmental scenarios. More specifically, this work develops an optimal motion planning framework that produces safe, natural-looking, and adaptive locomotive behaviors for several humanoid walking models as well as on a hardware system. The basis for this framework is an algorithmic approach that simultaneously decides in real-time where to place the legged systems' feet over a sequence of steps while determining how to dynamically coordinate its body to place its feet at the desired locations. In addition, we use a variant of differential dynamic programming to also simultaneously derive feedback controllers that stabilize the systems with respect to online disturbances.

The vast majority of conventional approaches to motion planning and control for humanoid robots moving through unstructured terrains make the assumption that the robot's motion over individual steps as well as the control thereof can be planned in

hierarchical stages: first footsteps, then motion, and finally control. However, in practice such disjoint approaches are often brittle in their application as unplanned motions, physical properties of the environment, dynamic obstacles, *etc.* often cause catastrophic failures. For example, consider a humanoid robot walking over uneven terrain. Imagine that the robot slightly slips as it is putting one of its feet on the ground, inducing a relatively large disturbance relative to some nominal gait. In this scenario, a conventional planning stack would first need to reason about how to execute a recovery motion by locating desired areas in the environment where the robot could brace itself or place its feet. Then, a motion planning routine would need to be executed to determine how to coordinate the robot's internal degrees-of-freedom to place its hands and/or feet at the desired locations. Finally, some low-level controller would need to be prescribed to stabilize the system during the online execution of this planned motion.

The issue with prior approaches is that they tend to either decouple kinematic and dynamic feasibility in treating footstep planning and motion planning separately; or they try to heuristically incorporate notions of dynamic feasibility into footstep planning. Each of these approaches can lead to sub-optimal behaviors and in the worst case total failure. In our approach, we explicitly consider the dynamic stability of the system when selecting where to place feet. In particular, we combine footstep planning, motion planning, and controller specification into a single algorithmic step.

More specifically, this work develops a novel approach for composing all three levels of planning and control (footsteps, motion, and feedback) concurrently by way of online quadratic programming. Our optimization decision points become where to place our feet, while indirectly solving the motion planning and control problems. We use ideas from nonlinear control theory to effectively discretize walking motions due to their inherently rhythmic nature. We then use the discretized dynamics within a differential dynamic programming (DDP) framework that requires a linearization of these dynamics. We show how to compute these linearized expressions, even though the nonlinear dynamics have this discontinuity, given knowledge of a discrete *impact map*. This map may be either specified analytically or learned through experimentation (e.g., on hardware). We

show how to embed terrain information into the framework to plan explicit motions for humanoid robots through contact over several step sequences.

Next, we present an informed energy-based template that allows our system to generate safe initial trajectories to be fed into the trajectory optimization pipeline as new terrain is perceived online. Using ideas from capture point theory, this method begins with determining whether the system has enough energy to carry itself over a specific foot impact. Through this energy-centric approach, we select trajectories that respect the natural "stability" of the system, even before considering the specific aspects of feedback control. By selecting initial trajectories using this method, we are indeed trying to select the best trajectories for both planning AND control, since the system should be easier to control if the actions are already aligned with natural dynamics of system. The energy technique becomes the building block to seed our unified trajectory optimization framework.

The resulting approach plans aggressive trajectories that avoid obstacles and safely adapts nominal walking motions to environments with significant mobility challenges. Finally, we bring all of these concepts together into one unifying architecture and demonstrate these techniques on various models and verify this technique on hardware.

1.1 High Level System Overview

The framework that we develop in this work unifies foot-step planning, motion planning, and feedback control by employing the closed-loop overall system's architecture shown in Fig. 1.1. The architecture is built around the classic "sense, plan, and act" paradigm. We realize this structure in three parallel processes.

Sense. The first process perceives the environment and builds a local terrain map using a vision system that is attached to the robot/robot model.

Plan. Second, the robot must determine how to physically traverse the terrain. A terrain cost generator is used to translate the terrain map into data structure that the robot can reasonably interpret within its planning framework. Because we are moving

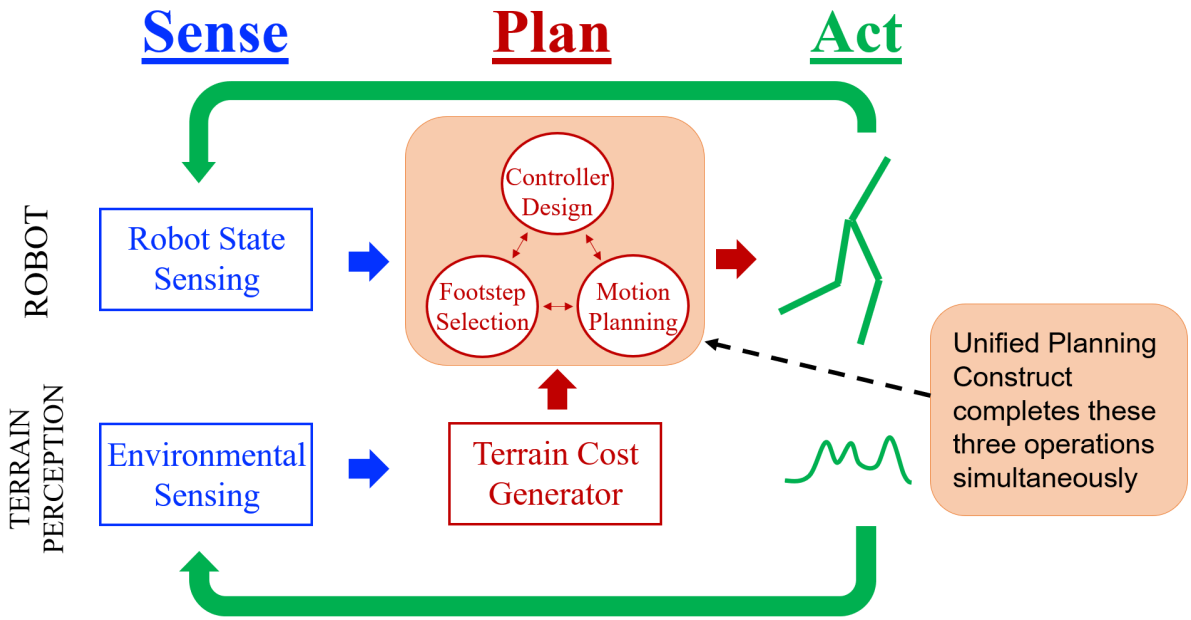

Figure 1.1: Unified Planning Architecture for a legged robot. This work focuses on footstep selection, motion planning and feedback controller design, all three simultaneously and in real-time.

dynamically over the terrain and may encounter cases where we are only able to see a short distance in front of us we need an approach that is highly agile and intuitive in design Therefore we need to plan where to step, how to move in between steps, and how to control the movement in real-time. By simultaneously planning these three elements in a model predictive control architecture, our unified planning construct enables real-time adaptation to terrain.

Act.

Finally, we execute the controller from the "plan." As we execute, we are already planning the next step.

1.2 Contributions

In this work, we have three contributions that are novel to the legged locomotion community:

1. We linearize our trajectory once-per-step at the moment in which the foot impacts the ground, thus embedding "where the foot comes down" into the linearized dy-

namics. By choosing to linearize at that specific point, we get foothold placement as a decisive byproduct of trajectory optimization.

2. This work derives two boundaries that are fundamental to walking: the first separates walking from falling while the second subdivides walking into slowing down and speeding up. We demonstrate how energy-centric foot placement can permit a passive system to move over rough terrain, rather than rejecting terrain through feedback.

3. We extend our energy technique to active systems in order to seed our trajectory optimization framework online. We then validate these methods through an implementation on hardware.

1.3 Organization of the Thesis

This thesis begins with a survey of the literature regarding robot walking and control in Chap. 2. We introduce hybrid dynamics and other important concepts in Chap. 3. We additionally use this chapter to derive all of the dynamic models used within this work. Chapters 4-6 contain the core of the contribution. We first introduce a novel method of unifying foothold selection, motion planning and feedback control into a single architecture. This work considers these three elements classically coupled and deliberately plans them simultaneously to generate bio-mimetic humanoid locomotive behaviors. We then recommend a technique of creating footstep and motion planning over rough terrain using a method of energy management. And then combine the two ideas to form an energy-based seed for the unified framework. We then show results for these methods in simulation and on hardware in Chap. 7. Lastly, we will present the conclusion, future work, and open problems in Chap. 8.

Chapter 2

Literature Survey

This work draws on a vast body of prior work on humanoid planning and control. We review relevant literature to provide context for the contributions presented throughout the rest of this document.

2.1 Hierarchical Planning and Control

Conventionally, motion planning approaches for legged systems break the problem down into three sub-problems: 1) The system surveys the terrain to identify candidate locations for places to place its feet, possibly over a sequence of several steps; 2) Then, a motion planning algorithm must determine how to generate dynamically-feasible paths between these locations; and, 3) Finally, a feedback controller is used to stabilize the system around the planned trajectories.

2.1.1 Footstep Selection

Footstep selection involves where and when to step, as shown in Fig. 2.1. Search-based methods are the most common approach found in the literature for solving the footstep location problem for legged robots. [1]–[7]. These inherently discrete methods first determine an initial set of footstep locations at its initial position, then determine footstep locations at the terminal position, and subsequently solve for number of intermittent locations to link them.

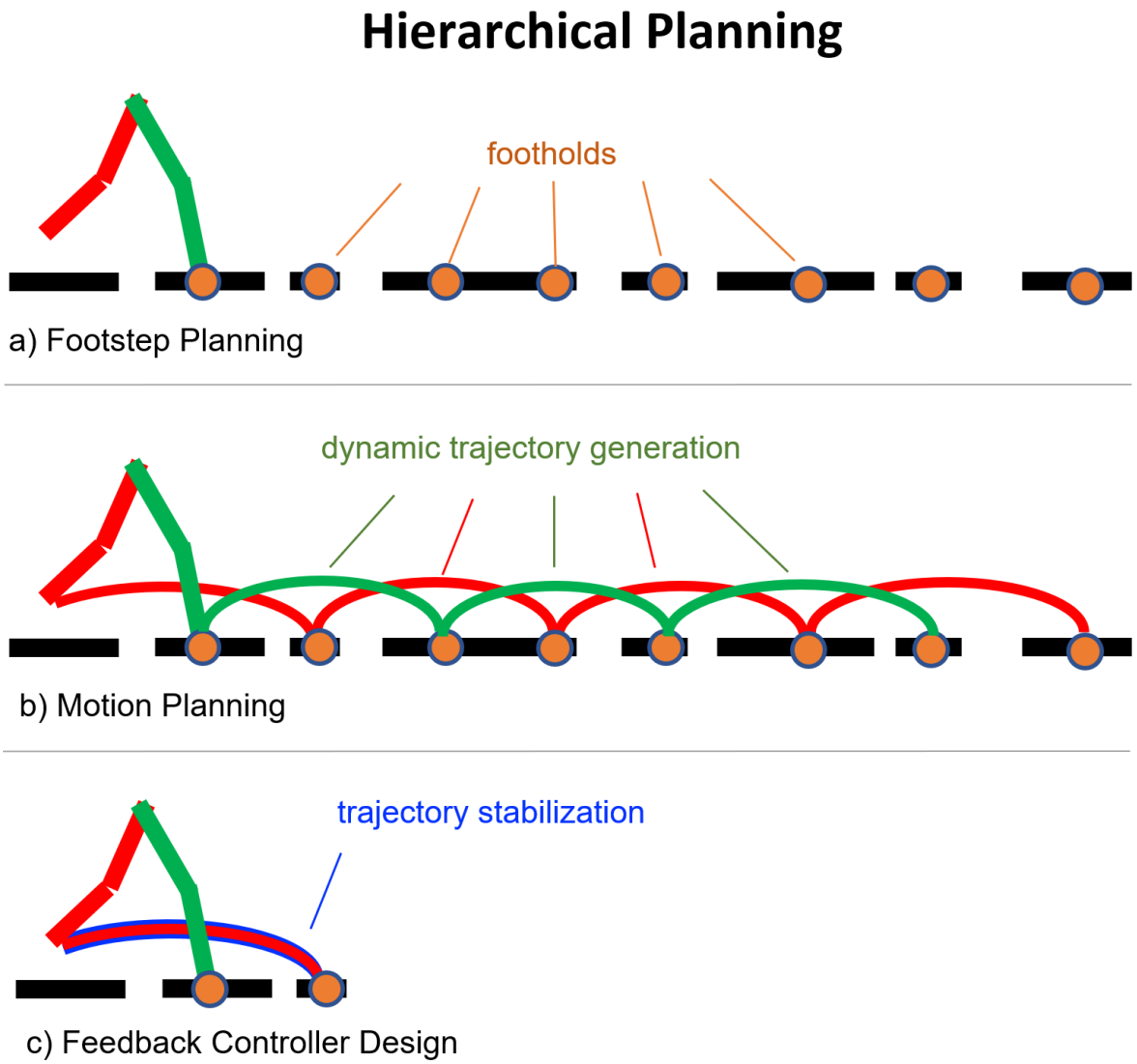

Figure 2.1: Hierarchical Planning. There are three classic levels of planning. a) The first action that occurs is footstep planning, which involves selection of where to place the feet. b) This is followed by motion planning, or determining the dynamic motion to transition between footholds. c) Finally, the trajectories must be stabilized along those trajectories to ensure disturbances are rejected and proper foothold selection occurs.

Search-based planners then typically find locally-optimal foothold positions by searching the *reachable space*, typically defined using the quasi-static stability margin of the system within a support polygon (centered within the vicinity of the nominal step). Furthermore, many previous works on footstep planning focus on systems that maintain kinematic stability, which self-limits the possible foothold locations, thereby limiting the types of terrain they can actually traverse. Six-legged and wheeled lunar rover ATHLETE [8], for instance, used a graph search to determine feasible and secure footholds,

and then, through inverse kinematics, chooses the leg positioning required to connect them [9]. Similarly implemented on the humanoid HRP-2 and quadraped Little Dog, [10] conducted a blind search of the kinematically-reachable area using A* to select the terrain of lowest cost and then used inverse kinematics to assign motion.

Researchers have extended footstep planning into dynamic motion using the idea of zero moment point (ZMP) [11] walking to humanoids such as ASIMO [12] and HOAP-2 [13]. ZMP is a point in which the dynamic reaction force with the ground does not produce a moment in the horizontal direction [14]. Stable walking is guaranteed by keeping the ZMP within the sole of the foot when in the single support phase or within the support polygon for double leg support [15]. Although ZMP walking provides more dynamic freedom than quasi-static walking, the selection of gaits are still limited. For example, ZMP considers a constant COM height linear inverted pendulum (discussed in Chap. 3), which allows researchers to use many forms of very fast and efficient linear tools, yet considerably constricts the possible motion of the system.

Other dynamic approaches [16], [17] utilize offline generation of a library of motion primitives that link states via optimized trajectories. Then, online, efficient search tools can select the foothold positions that correspond to motion within their primitive library. Though efficient in real-time planning, the size of the library grows exponentially with the size of the state space and they can be difficult to adapt in complex terrains due to a lack of generalizability.

Once the footstep plan culminates, the hierarchical approaches proceed on to the motion-planning and controller derivation problems.

2.1.2 Motion Planning

Motion planning involves solving how to step at these foothold locations, as indicated in Fig. 2.1. A number of continuous footstep and motion planning approaches have recently been formulated as trajectory optimization problems [4], [18]–[20]. This is done by using a direct collocation technique where the points on the state and control trajectory along with other parameters form the decision variables. The main advantage of this

class of approaches is that the dynamics of the system can be directly incorporated into the motion planning portion of the approach. Prior related approaches [21] work in near-real-time, but require global maps of the environment to be known ahead of time (analytical solutions to the Jacobians of the constraints were prepared and provided to the solver). In the formulation of this work, the terrain is unstructured and and therefore this methodology is not sufficient for online planning as new terrain information is gained.

Additionally, these first two stages of planning can happen in reverse. ATHLETE, for instance, uses a two-stage search strategy to determine footstep locations and strategies, both remaining hierarchical [9], [22]. 1) if the terrain is permissive and the system presumed stable, the planner chooses the body motion motion using a fixed gait before later planning the exact footfalls and desired motion to keep the COM stable [23]. 2) If terrain is irregular or steep and contact stability is important, the system first chooses footfalls before choosing the required quasi-static motion to complete the steps.

It is important to note that motion planning is either developed using reduced order dynamic models (ROMs) (e.g. planning for system COM, using model simplifications such as inverted pendulum, *etc.*) or full-body dynamic models (complete joint planning). Few footstep and motion planners incorporate full-body models, which complicates feedback control.

2.1.3 Feedback Control

As the final part of the hierarchy shown in Fig. 2.1, the motion plan is then sent to the controller for stabilization of the robot motion. If the motion planner considered only a ROM, an additional layer of control must also now be considered. First, a high level controller commands motions of specific body parts (e.g. hands, foots, and body) in cartesian space. Next, a low level controller inputs the workspace motion and outputs joint level controllers that control angular position, velocity, and torque. Atlas, for instance, used this hierarchical control structure in the DARPA Robotics Challenge [24]–[26].

Skipping the two-step approach, some modern approaches consider the joint level control requirements directly in their motion planning. [27], [28] add joint level state and

control constraints to their planner by combining control barrier functions and control Lyapunov functions to the severely underactuated robot RABBIT [29]. This approach allows the system to adjust motion and control foot placement in real-time, while respecting the lower level control and state constraints. In another example, [30] pre-computed a library of controllers about individual gaits for ATRIAS [31], enabling the robot robot to react to disturbances in way that is tailored to that specific stepping motion.

Either way of parsing the controller, these robots require whole body control in order to dynamically stabilize under uncertainty. We use the latter approach, where a full order dynamic model is passed to the motion planner for optimization. We believe this yields the strongest controller, where control is tuned to the full motion of the step rather than compensating for error in the ROM.

2.2 Combining Footstep Selection, Motion Planning, and Feedback Control

This hierarchical approach creates a loose coupling between planning footholds and the underlying policy that executes the motion to carry out the footsteps. A dynamic motion policy might underperform given a conservative footstep planner, while execution might completely fail given an aggressive planner. Therefore, prior works have combined some elements of footstep selection, motion planning and feedback control.

2.2.1 Combined Footstep Selection and Motion Planning

A novel approach to deconflict the hierarchical aggressiveness is through the passing of certificates [32] that correspond to a level of agreement between the footstep plan and the capabilities of the motion planners. These algorithms then use a local, gradient-based search across the certificates to determine where to place feet in real-time. Utilizing the trajectory optimization software CHOMP [33], the method was successfully implemented on Little Dog in real-time during the DARPA Learning Locomotion project, yet this architecture still assumes *a priori* terrain knowledge and cost and prior computation

candidate actions.

Recently a number of approaches [4], [19], [20] have combined footstep selection and motion planning into trajectory optimization formulations by directly incorporating the dynamics into the footstep planning. [21] showed the approach can work in near-real-time, but require a map of the environment to develop a phase-based parameterization using analytical representations of the holonomic constraints imposed by the environment. While these methods can yield both foothold and motion plans, they still do not consider the amount of feedback control effort it takes to control around the motion plan.

Other planning methods [34], [35] discount discrete impulses at contact, assuming the contact forces are exactly the forces required to maintain the foot in contact by interacting with the environment through inelastic impacts and Coulomb friction. This allows the planner to discount the hybrid nature of walking and consider only continuous phases, allowing run-time speedups by direct sequential quadratic programming solvers such as SNOPT [36] that require smooth functions. Although their methods require that the dynamics remain continuous when passing through contact, we will show how we can preserve the discontinuous nature of contact through our unified method.

2.2.2 Combined Planning and Control

Differential Dynamic Programming (DDP) [37] is a class of approaches that natively produces not only the trajectory, but also the controller to track them. These methods create a value function consisting of quadratic approximations of a cost-to-go function and dynamics around a trajectory. A minimization of the value function is then used to make incremental updates to the control during iterative several forward and backward passes, in order to solve for locally-optimal trajectories and the controllers to stabilize them.

To incorporate environmental information into their numerical optimization, prior works demonstrated the ability to plan with pure-state and state constraints using DDP variants [6], [38]–[41]. This is done by incorporating the quadratically-weighted linearized state-input constraints into their cost function by using Lagrange multipliers [42], [43].

In all of these methods, undesirable states are modelled as barrier-like constraints, i.e. they are explicitly avoided. By modeling the hybrid legged robot as a switched system, [44] extended hard-constrained DDP to legged locomotion, but additionally solves for switching time parameters, a step unnecessary in our formulation. Current optimization-based methods [20], [41] directly use the terrain height map as a constraint on the state with the use of Lagrange multipliers.

Closely related to DDP is the iterative linear quadratic regulator (iLQR) [45], [46], which has presented promising results in terms of solving planning problems in real-time [47]. Instead of making quadratic approximations about the dynamics, the approximations are linear creating a computationally simpler algorithm at the cost of a slightly degraded local approximation. In fact, [45] showed that the DDP second order approximations of the dynamics were computationally expensive and potentially inaccurate. In a series of reasonably complex control problems, they showed that iLQR turned out to be more efficient by a factor of 10, recently enabling real-time applications for more complex systems [47].

Our work in [48] is most closely related to [41], where they use a DDP variant called Sequential Linear Quadratic (SLQ) programming in order to directly solve for the optimal time of contact t_c for a series of steps. Their method overcomes the hurdle of t_c changing between iterations, which plagues most of the methods above since DDP handles continuous phases as a time-varying tracking problem. Our method also solves for t_c, yet indirectly. Though, their approach segments each swing phase into many pieces while our method projects the entire step onto a lower dimensional surface for optimization. This work uses an SLQ formulation as the basis for our planner, in which we are able to simultaneously solve for foothold locations, dynamically feasible trajectories, and controllers that stabilize around that trajectory as shown in Fig. 2.2.

In this work, we differentiate these iterative variants as follows:

- **DDP**: value function consists of 2nd order (quadratic) approximation of both cost function and dynamics
- **SLQ**: value function consists of 2nd order approximation of cost function but 1st

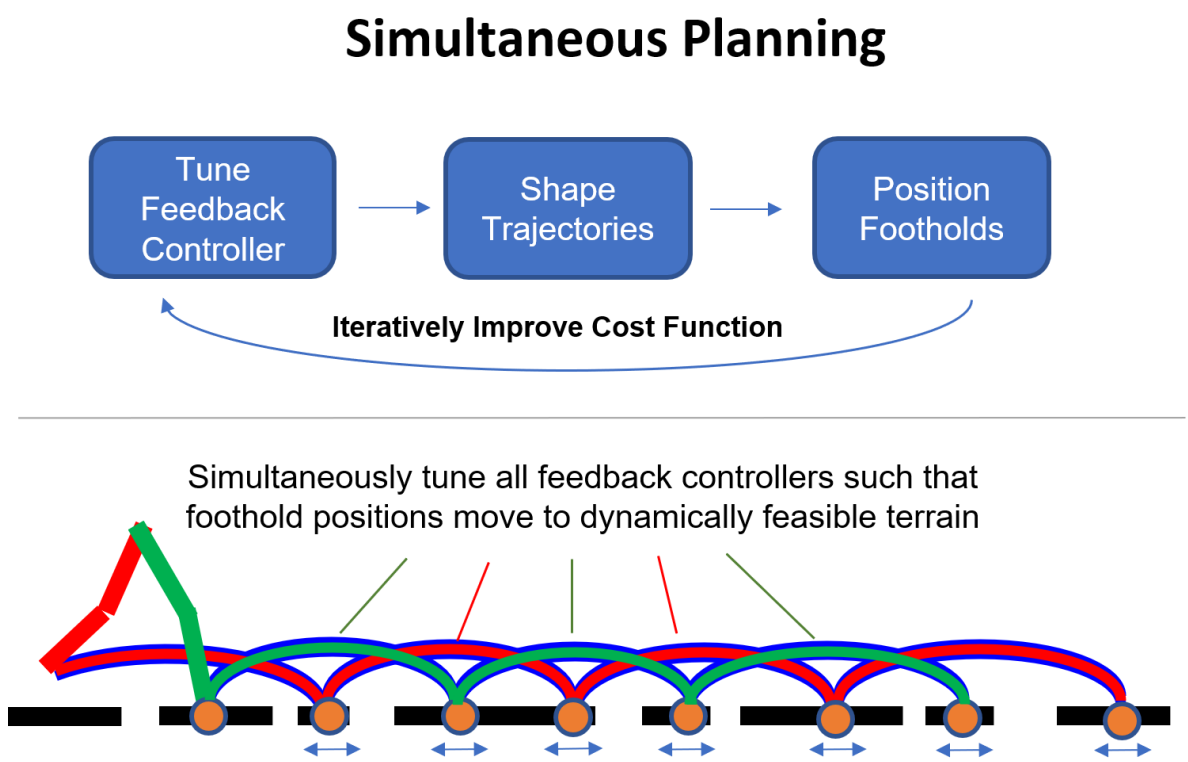

Figure 2.2: Simultaneous Planning. All three levels of planning are combined into one process. In this work, feedback controllers are adjusted such that the footholds conform to the terrain using dynamically feasible trajectories. This planning process occurs once per footstep, using a receding planning horizon under model predictive control.

order (linear) approximation of dynamics

- **iLQR**: value function consists of 1st order approximation of both cost function and dynamics

2.3 Generating an Initial Guess to Seed the Planner

DDP is fundamentally a local optimization method, so the quality of the optimized plan depends very directly on the quality of the chosen seed. Many works [47], [49]–[51] start a DDP algorithm with the requirement to initiate with a feasible trajectory, but do not offer methods as to how to generate the seed. This can be challenging over uneven terrain and varying terrain heights for foot contact.

Additionally, the seed for one hybrid step must end at the beginning of the next hybrid step. Considering an underactuated walking system with no ankle torque, many

of the interim steps are likely to result in falling over if footsteps are not carefully stitched together. Therefore, creating such an initial trajectory (consisting of several new steps) creates another layer of planning that we refer to as *seeding*.

There are two common means to generate initial seeds for optimal motion planning algorithms such as DDP and iLQR. The first is to use the aforementioned search-based, feasible planning method to sample the state space. The second would be to utilize a constraint to limit the seed to a kinematic search area that we know fits within the dynamic construct. We approach the latter using a method of energy management based on the idea of capture point [52]. Energy and capture point are intrinsically linked, and together provide useful insight for generating walking motion.

2.3.1 Energy Management

There is a long history of energy-based analysis using passive dynamic walking systems. One of the greatest strides in biped walking came forth in the early 1990s when McGeer introduced the passive kneed walker [53], [54], a system of two interacting pendula using gravity as a means of compelling forward motion. The system balanced energy through an interplay of ground contacts (kinetic energy lost) and traveling down a slope (potential energy gained), starting a trend of passive walking analysis [55]–[62]. This demonstrated that there is grace, efficiency and robustness to simplicity and balance, and that walking does not always require control. In a similar way, this work takes a step back to the passive nonlinear inverted pendulum (NIP) model walking down a hill, and then later applies the concept to that of controlled walking systems walking over various terrain.

2.3.2 Capture Point

Based on the dynamics of a linear inverted pendulum (LIP), *capture point* (CP) [52] was developed as a means of controlling a force disturbance. A CP is a terrain location that the robot can step in order to come to a complete stop. Although the idea of of CP can apply to all forms of dynamic walking, researchers have used the concept primarily around the simplified dynamics model of the LIP during ZMP walking. The dynamic equivalent

of static walking, ZMP walking maintains the ZMP within the support polygon, but allows the floor projected COM to deviate during a gait. For a known COM trajectory, the CP is very fast to calculate and therefore very useful in push recovery [52], [63]–[69], i.e. step on CP to cease movement. We use notion of CP in a related way to prior works, but in a unique way that enables us to intelligently create multi-step seeds to be passed off to our unified planning framework discussed in Chap. 4.

Chapter 3

Hybrid Dynamic Walking Models

For over thirty years robotics researchers have used simple simple, reduced-order walking models that exemplify the salient features of biological walking but are much more tractable to study [20], [54], [55], [70]–[72] than biologically-inspired designs that are true to the motion of living creatures. In this chapter we introduce the basic assumptions as well as specific walking models we use throughout the remainder of this work.

3.1 Hybrid Dynamics

For bipedal models with point feet, we assume there are two phases in a nominal step of a walking gait. In the first, the robot pivots about a single supporting leg while its other swings forward toward its next position on the ground. In the second phase the robot is in double support, wherein both of the robot's feet are in contact with the ground at the same time, as shown in Fig. 3.1. This phase serves as the transition of the previous stance leg becoming the new swing leg, and vice-versa. When both phases are complete (e.g. left foot was swing for the last phase but is now stance for the upcoming phase, and vice versa for the right foot), a step is complete and the cycle continues.

In this work, the double support phase is considered to be instantaneous, occurring at the time the leg in swing impacts the ground. Analytically, we assume that this instantaneous transfer of support defines a discrete mapping M from the state immediately

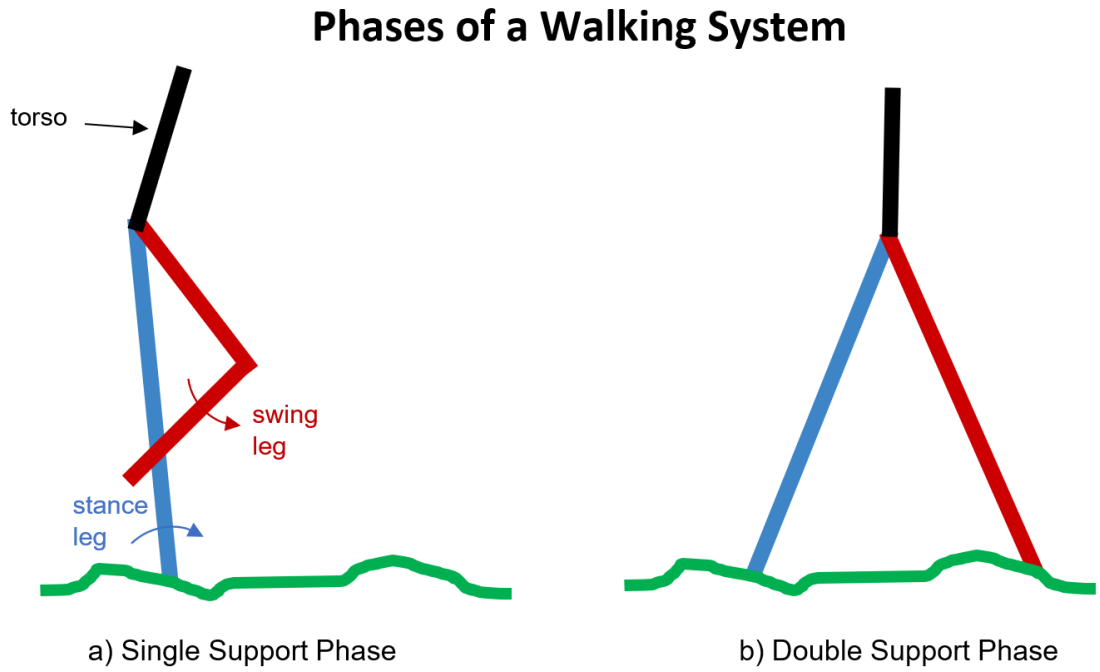

Figure 3.1: Two Phases of a walking system. (a) Single support or swing phase. (b) Double support phase. NOTE: A third aerial phase is not included in this work as we do not consider running.

before collision $\mathbf{q}^-_{[n]}$ to the state immediately after collision $\mathbf{q}^+_{[n]}$

$$\mathbf{q}^+_{[n]} = M\left(\mathbf{q}^-_{[n]}\right), \tag{3.1}$$

where n represents the discrete step number. The form of state $\mathbf{q}$ used within this work is defined in Sect. 3.1.1. The impact map M represents the change in the system's state through the collision with the ground. In this work, we use the conservation of angular momentum [73] to define M.

The swing phase consists of continuous dynamics $f(\mathbf{q}(\mathrm{t}), \boldsymbol{\tau}(\mathrm{t}))$ swinging from post-collision state $\mathbf{q}^+_{[n]}$ until the time of the next collision t_c of the new swing foot

$$\begin{aligned}\mathbf{q}\left(\mathrm{t_c}\right) &= \mathbf{q}^+_{[n]} + \int_0^{\mathrm{t_c}} f(\mathbf{q}(\mathrm{t}), \boldsymbol{\tau}(\mathrm{t}))\,\mathrm{dt} \\ &= \mathbf{q}^-_{[n+1]},\end{aligned} \tag{3.2}$$

where state $\mathbf{q}$ and control $\boldsymbol{\tau}$ are explicitly dependent upon time. This pre-collision state $\mathbf{q}^-_{[n+1]}$ marks the next step as shown in Fig. 3.2. Combining (3.1) and (3.2), yields the

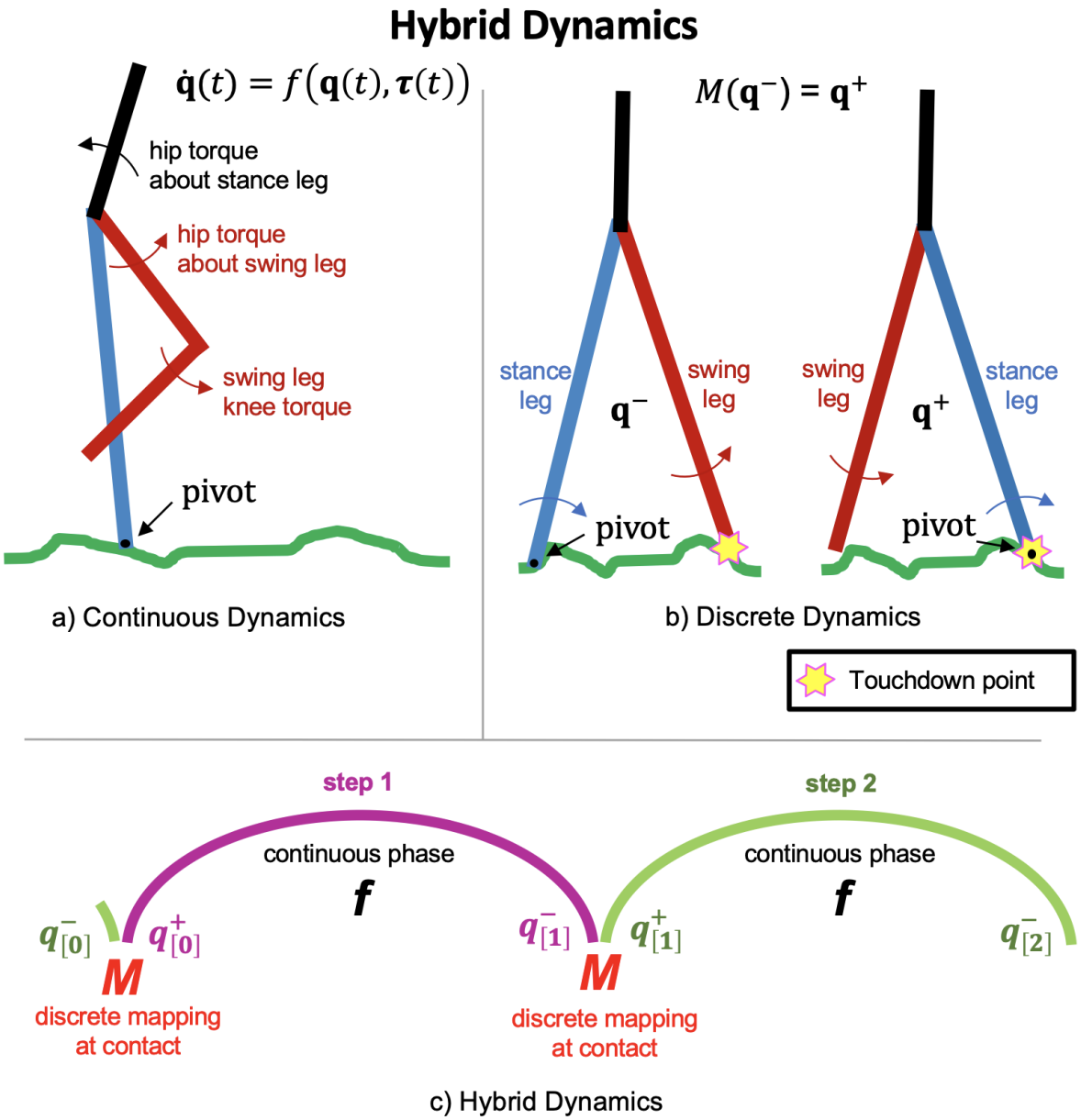

Figure 3.2: Hybrid Dynamics. This image shows a step consisting of two phases. (a) Continuous motion with dynamics f that propagate the state space $\mathbf{q}$(t) during the swing phase. (b) A discrete mapping M at impact carries the state from a precollision-state $\mathbf{q}^-$ to a post-collision state $\mathbf{q}^+$. The pivot point changes to the the stance foot.

hybrid dynamics

$$\mathbf{q}^-_{[n+1]} = M\left(\mathbf{q}^-_{[n]}\right) + \int_0^{t_c} f(\mathbf{q}(t), \boldsymbol{\tau}(t))\, dt. \tag{3.3}$$

In addition, similarly to [52], [72], [74]–[77], we make the following assumptions:

1. Only one leg is in contact with the ground at a time (no double support phase).

2. Ground impact is inelastic and the system does not rebound/bounce upon impact.

3. The stance foot does not slip and acts as an uncontrolled pin joint (some of the referenced papers use friction models and/or ankle control).

3.1.1 Hybrid Swing Phase - Continuous Dynamics

During swing a planar walking system, this work uses the standard robot manipulator equations of motion

$$\mathbf{M}(\mathbf{\Theta})\ddot{\mathbf{\Theta}} + \mathbf{C}(\mathbf{\Theta}, \dot{\mathbf{\Theta}})\dot{\mathbf{\Theta}} + \mathbf{g}(\mathbf{\Theta}) = \boldsymbol{\tau}. \tag{3.4}$$

For an S degree of freedom (S-DOF) system, $\mathbf{\Theta} = \begin{bmatrix} \theta_1 & \dots & \theta_S \end{bmatrix}^{\mathrm{T}}$ represents the vector of joint angles and $\dot{\mathbf{\Theta}}$ and $\ddot{\mathbf{\Theta}}$ represent the first and second time derivatives, respectively. $\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \dots & \tau_S \end{bmatrix}^T$ represents a $S \times 1$ vector of joint torques. Within this work, it is always assumed that the stance ankle is unactuated, therefore $\tau_1 = 0$. $\mathbf{M}(\mathbf{\Theta})$ is the $S \times S$ inertia matrix, $\mathbf{C}(\mathbf{\Theta}, \dot{\mathbf{\Theta}})$ is the $S \times S$ coriolis matrix, and $\mathbf{g}(\mathbf{\Theta})$ is the $S \times 1$ gravity-compensation vector. The values of $\mathbf{M}, \mathbf{C}, \mathbf{g}$ can be systematically derived from the Euler-Lagrange equations in a recursive set of equations shown in [78], [79] using only the Denavit-Hartenberg (DH) parameters [80] and the mass properties. For each S-link walking system used within this work, the DH parameters and mass properties will be provided.

For the planar manipulator shown in Fig. 3.2, we additionally need to keep track of translational motion, which is captured as translation x and height z of the stance foot which remains constant during the phase. Although the translational movement x, z does not affect the continuous dynamics, it is necessary in our full hybrid mapping as well as tracking the swing foot position relative to the terrain. The robot can now be

fully represented by the $(2S+2)\times 1$ state space vector $\mathbf{q}$

$$\mathbf{q} = \begin{bmatrix} \boldsymbol{\Theta} \\ \dot{\boldsymbol{\Theta}} \\ x \\ z \end{bmatrix}.$$

Therefore, the continuous dynamics f are

$$f(\mathbf{q}, \boldsymbol{\tau}) = \dot{\mathbf{q}} = \begin{bmatrix} \dot{\boldsymbol{\Theta}} \\ \mathbf{M}(\boldsymbol{\Theta})^{-1}\left(\boldsymbol{\tau} - \mathbf{C}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})\dot{\boldsymbol{\Theta}} - \mathbf{g}(\boldsymbol{\Theta})\right) \\ 0 \\ 0 \end{bmatrix}. \tag{3.5}$$

3.1.2 Hybrid Collision Map - Discrete Dynamics

When the swing foot eventually touches the ground, signaling the transition to double support, there is a discontinuous jump in the state $\mathbf{q}$ due to the impulse the swing foot experiences when contacting the environment. At this instant the swing foot becomes the stance foot, and vice versa, as shown in Fig. 3.3.

We use an $S \times S$ configuration matrix $\mathcal{Q}_{\mathbf{q}}$ that relates the pre-collision configuration to the post-collision configuration

$$\boldsymbol{\Theta}^{+} = \mathcal{Q}_{\mathbf{q}}\left(\boldsymbol{\Theta}^{-}\right). \tag{3.6}$$

The world coordinates of the end effector (swing foot) at touchdown can be determined by the configuration of the joints and the stance foot global coordinates x^{-}, z^{-}. We assume there to be a pin joint at this touchdown location, which becomes the new stance foot reference for the succeeding continuous phase

$$\begin{bmatrix} x \\ z \end{bmatrix}^{+} = \mathcal{Q}_{\mathbf{xz}}\left(\mathbf{q}^{-}\right). \tag{3.7}$$

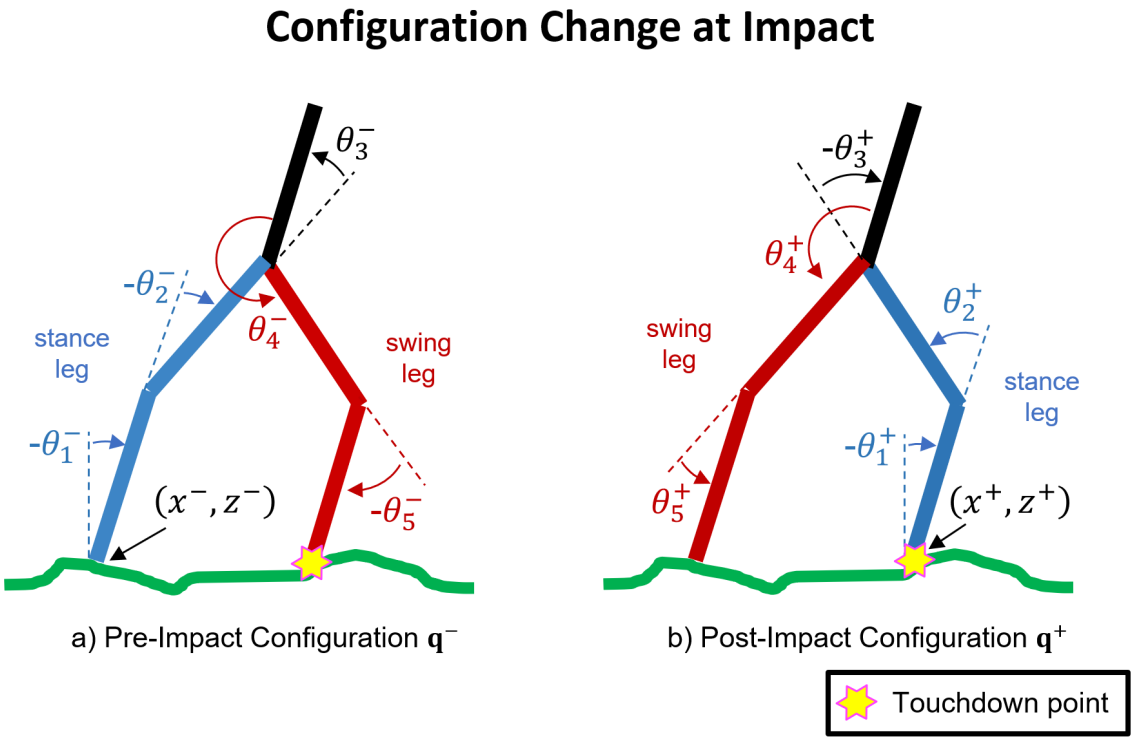

Figure 3.3: Configuration Change at Impact. (a) Pre-Impact Configuration $\mathbf{q}^-$ (b) Post-Impact Configuration $\mathbf{q}^+$

Finally, we need to determine the relationship between the pre- and post-collision joint velocities, $\dot{\mathbf{\Theta}}^-$ and $\dot{\mathbf{\Theta}}^+$, respectively. The double support phase is assumed instantaneous as mentioned in Sect. 3.1, which means that the internal joint configuration does not change during collision and the external forces from the ground must be impulsive in nature. Since the only external force on the biped is the impact force at the foot, the angular momentum about the touchdown foot is conserved throughout impact. The angular momentum L is determined by summing its center of mass (COM) momentum and the momentum of the inertia about that COM.

$$L_s = \sum_{a=1}^{s} \mathbf{r}_{\mathrm{a}} \times \mathrm{m}_{\mathrm{a}} \mathbf{v}_{\mathbf{a}} + j_{\mathrm{s}} \dot{\theta}_{\mathrm{a}} \tag{3.8}$$

where $\mathbf{r}_{\mathrm{a}} = \mathbf{r}_{\mathrm{com_a}} - \mathbf{r}_{\mathrm{TD}}$, and

$$\mathbf{v}_{\mathrm{a}} = \sum_{\mathrm{b}=1}^{\mathrm{a}-1} \left[l_{\mathrm{b}} \left(\sum_{\mathrm{d}=1}^{\mathrm{b}} \dot{\theta}_{\mathrm{d}} \right) \begin{bmatrix} -\cos\left(\sum_{d=1}^{b} \theta_d\right) \\ -\sin\left(\sum_{d=1}^{b} \theta_d\right) \\ 0 \end{bmatrix} \right] + l_{c_{\mathrm{a}}} \left(\sum_{\mathrm{d}=1}^{\mathrm{a}} \dot{\theta}_{\mathrm{d}} \right) \begin{bmatrix} -\cos\left(\sum_{d=1}^{a} \theta_d\right) \\ -\sin\left(\sum_{d=1}^{a} \theta_d\right) \\ 0 \end{bmatrix}.$$

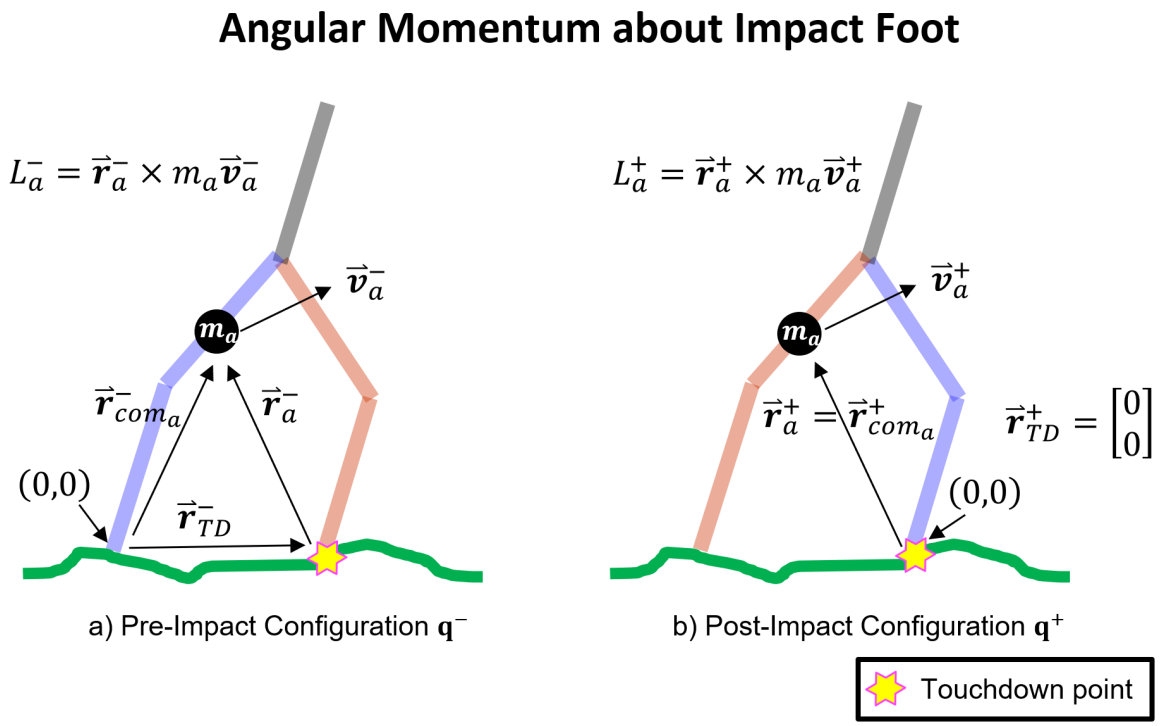

Figure 3.4: Angular Momentum about Impact Foot. (a) Pre-Impact Configuration $\mathbf{q}^-$. All terms are written in terms of $\mathbf{\Theta}^-$, $\dot{\mathbf{\Theta}}^-$, x^-, and z^-. (b) Post-Impact Configuration $\mathbf{q}^+$. All terms are written in terms of $\mathbf{\Theta}^+$, $\dot{\mathbf{\Theta}}^+$, x^+, and z^+.

The vector $\mathbf{r}_{\text{com}}$ represents the position of the center of mass (COM) of link a and vector $\mathbf{r}_{\text{TD}}$ represents the position vector of the end effector at touchdown in the stance foot frame. The vectors for the a^{th} "point mass" are shown in Fig. 3.4. The link lengths l_i and length to the link COM l_{c_i} are scalar distances in the link frame. The link moment of inertia about its COM for planar rotations is annotated by scalar j_s.

The link angular momentum L_s in (3.8) can be represented by the linear relationship

$$L_s = \mathcal{Q}_{\mathbf{q}_s} \dot{\mathbf{\Theta}},$$

where the $1 \times S$ vector $\mathcal{Q}_{\mathbf{q}_s}$ contains the configuration and mass terms for the s^{th} link.

The set of all link momenta can be represented by the $S \times 1$ vector $\mathbf{L}$

$$\begin{bmatrix} L_1 \\ \vdots \\ L_S \end{bmatrix} = \begin{bmatrix} \mathcal{Q}_{\dot{\mathbf{q}}_1} \\ \vdots \\ \mathcal{Q}_{\dot{\mathbf{q}}_S} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_S \end{bmatrix}$$
$$\mathbf{L} = \mathcal{Q}_{\dot{\mathbf{q}}}(\boldsymbol{\Theta})\,\dot{\boldsymbol{\Theta}},$$

where $\mathcal{Q}_{\dot{\mathbf{q}}}$ is the $S \times S$ configuration matrix for the system of S links about the impact point. The pre- and post-collision angular momenta about the impact point can be written as $\mathbf{L}^-$ and $\mathbf{L}^+$, respectively.

For an inelastic collision, the force at touchdown is the only external force on the system. Since the touchdown force is a vector through the pivot point, it has no bearing on angular momentum about that point.

Therefore for a rigid body, we can determine that the angular momentum about that new pivot is preserved through collision. We can then use this equivalence to solve for the post collision-velocity $\dot{\boldsymbol{\Theta}}^+$

$$\begin{aligned} \mathbf{L}^+ &= \mathbf{L}^- \\ \mathcal{Q}_{\dot{\mathbf{q}}+}\left(\boldsymbol{\Theta}^+\right)\dot{\boldsymbol{\Theta}}^+ &= \mathcal{Q}_{\dot{\mathbf{q}}-}\left(\boldsymbol{\Theta}^-\right)\dot{\boldsymbol{\Theta}}^- \\ \dot{\boldsymbol{\Theta}}^+ &= \left[\left(\mathcal{Q}_{\dot{\mathbf{q}}+}\left(\boldsymbol{\Theta}^+\right)\right)^{-1}\mathcal{Q}_{\dot{\mathbf{q}}-}\left(\boldsymbol{\Theta}^-\right)\right]\dot{\boldsymbol{\Theta}}^- \end{aligned} \tag{3.9}$$

Combining (3.6) and (3.9) makes it possible to rewrite the post-collision velocity in terms of the pre-collision states only

$$\begin{aligned} \dot{\boldsymbol{\Theta}}^+ &= \left[\left(\mathcal{Q}_{\dot{\mathbf{q}}+}\left(\boldsymbol{\Theta}^-\right)\right)^{-1}\mathcal{Q}_{\dot{\mathbf{q}}-}\left(\boldsymbol{\Theta}^-\right)\right]\dot{\boldsymbol{\Theta}}^- \\ &= \mathcal{Q}_{\dot{\mathbf{q}}+}^{-1}\mathcal{Q}_{\dot{\mathbf{q}}-}\dot{\boldsymbol{\Theta}}^- \end{aligned} \tag{3.10}$$

At impact, the pre-collision states $\mathbf{q}^-$ can be transitioned to post-collision states $\mathbf{q}^+$

using (3.6), (3.10), and (3.7)

$$\begin{bmatrix} \boldsymbol{\Theta} \\ \dot{\boldsymbol{\Theta}} \\ x \\ z \end{bmatrix}^+ = \begin{bmatrix} \mathcal{Q}_{\mathbf{q}} \\ \mathcal{Q}_{\dot{\mathbf{q}}^+}^{-1} \mathcal{Q}_{\dot{\mathbf{q}}^-} \dot{\boldsymbol{\Theta}} \\ \mathcal{Q}_{\mathbf{xz}} \end{bmatrix}^- \tag{3.11}$$

$$\mathbf{q}^+ = M\left(\mathbf{q}^-\right). \tag{3.12}$$

3.2 Planar Walking Systems

This work considers several different models for dynamic walking. This section breaks out three models, each increasing in complexity. The first considers a passive centroidal system with rigid, massless legs. The second adds masses to the legs, creating the requirement to actively swing a massive leg during the continuous swing phase. The final model, which we use on our hardware, adds knees and a torso to the model, creating a more comprehensive biped.

3.2.1 Nonlinear Inverted Pendulum (NIP) Model

The simplest of these models is a point mass with mass-less legs. Many prior works have investigated this system given the assumption that they are able to maintain the COM at a constant height z [52], [81] (i.e. $\dot{z} = \ddot{z} = 0$). In this case, the system's dynamics in each swing phase become linear, resulting in the well known LIP model for walking as per Fig. 3.5. Though relatively simple, the LIP model has been enormously successful in prior works on bipedal walking [12], [82]–[89]. A key assumption for the LIP model are that impact is smooth and continuous, creating no discrete events or hybrid dynamics.

The NIP model handles walking motion instead with a rigid stance leg, shown in Fig. 3.5. Although still a simple system, the stiff leg creates an arcing motion and complicates the dynamics such that they are no longer linear. Note, the system COM is no longer restricted to the transverse plane ($\dot{z} \neq 0$). The swing leg impacts the

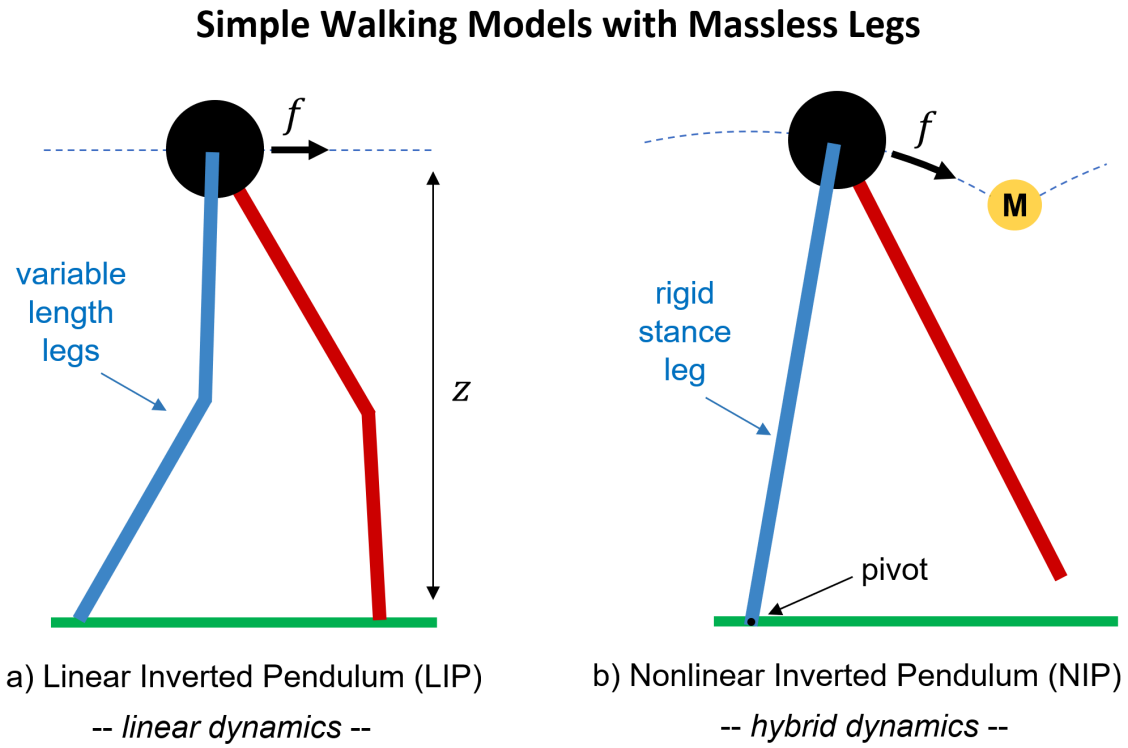

Figure 3.5: Simple Walking Models with Massless Legs. a) The legs of the LIP model provide exactly the vertical force ($F = mg$) necessary to maintain the system COM at a constant height z. The leg positioning and force control are assumed to be a problem for a lower level controller to regulate. The system smoothly moves through elastic contact with the ground, providing completely linear dynamics. b) The rigid system pivots about the stance foot, moving the COM along an arc with nonlinear dynamics f. At the moment of contact, the system loses energy through a discontinuous, inelastic collision M. This system has hybrid dynamics.

ground with a non-conservative impact, wherein kinetic energy is lost from the system and an instantaneous change in pivot point occurs as discussed in Sect. 3.1.2. This work considers a passive NIP system with massless legs (see Fig. 3.6) walking downhill whose DH parameters and mass properties are given in Table 3.1.

Table 3.1: Denavit-Hartenberg Parameters and Mass Properties for Nonlinear Inverted Pendulum Walker with Massless Legs

Link	a_i	α_i	d_i	θ_i	m_i	l_{c_i}	j_i	Description
1	l	0	0	θ_1	m	l	0	stance
2	l	0	0	θ_2	0	0	0	swing

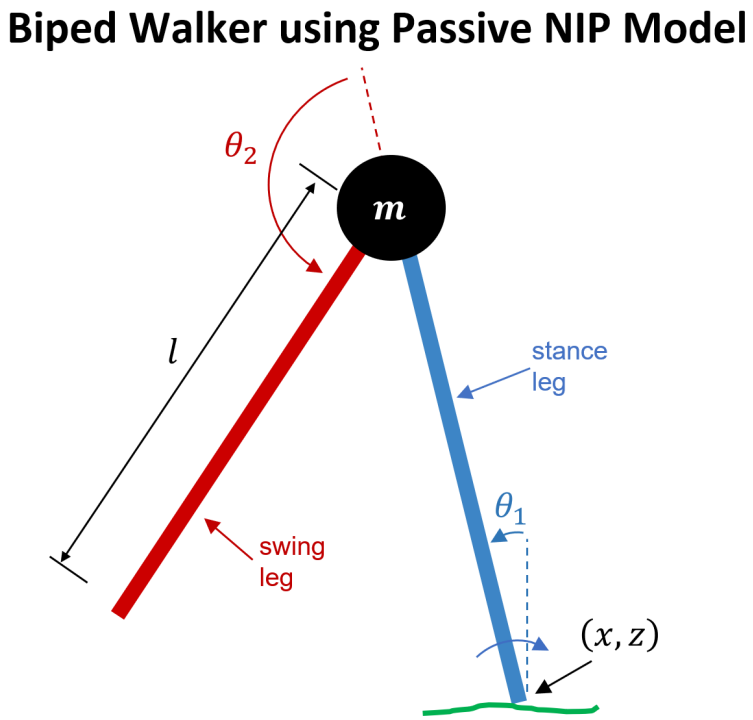

Figure 3.6: Biped Walker using the Nonlinear Inverted Pendulum Model. Passive system with point mass at the hip and a pin joint at the stance foot.

The dynamics can be given by (3.4) and

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} ml^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.13}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0} \tag{3.14}$$

$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix} mgl\sin(\theta_1) \\ 0 \end{bmatrix}, \tag{3.15}$$

and $\tau_2 = 0$. Following (3.11), the impact map can be given by

$$\mathcal{Q}_{\mathbf{q}} = \begin{bmatrix} \theta_1 + \theta_2 - \pi \\ -\theta_2 \end{bmatrix}^- \tag{3.16}$$

$$\mathcal{Q}_{\dot{\mathbf{q}}^+} = \begin{bmatrix} ml^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.17}$$

$$\mathcal{Q}_{\dot{\mathbf{q}}^-} = \begin{bmatrix} ml^2 \cos\theta_2 & 0 \\ 0 & 0 \end{bmatrix}^- \tag{3.18}$$

$$\mathcal{Q}_{\text{xz}} = \begin{bmatrix} x - l\left(\sin\theta_1 + \sin(\theta_1 + \theta_2)\right) \\ z + l\left(\cos\theta_1 + \cos(\theta_1 + \theta_2)\right) \end{bmatrix}^- . \tag{3.19}$$

We assume that the swing leg is massless and therefore its velocity $\dot{\theta}_2$ does not appear in the formulation of (3.13)-(3.18). This leads to a simplified form of the continuous and discrete dynamics given by

$$f(t) = \begin{bmatrix} \dot{\theta}_1(t) \\ \frac{g}{l}\sin\left(\theta_1(t)\right) \\ 0 \\ 0 \end{bmatrix} \tag{3.20}$$

$$\mathbf{q}^+ = M(\mathbf{q}^-)$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ x \\ z \end{bmatrix}^+ = \begin{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & \cos\theta_2 \end{bmatrix}^- \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \end{bmatrix}^- + \begin{bmatrix} -\pi \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} -\sin\theta_1 - \sin(\theta_1 + \theta_2) \\ \cos\theta_1 + \cos(\theta_1 + \theta_2) \end{bmatrix}^- l + \begin{bmatrix} x \\ z \end{bmatrix}^- \end{bmatrix} . \tag{3.21}$$

Since the swing leg is considered massless and can be placed arbitrarily, the inter-leg angle θ_2 does not appear in the continuous dynamics (3.20), yet its pre-impact configuration does appear in the impact map (3.21). Therefore, $\dot{\theta}_2$ and $\ddot{\theta}_2$ are excluded from the hybrid

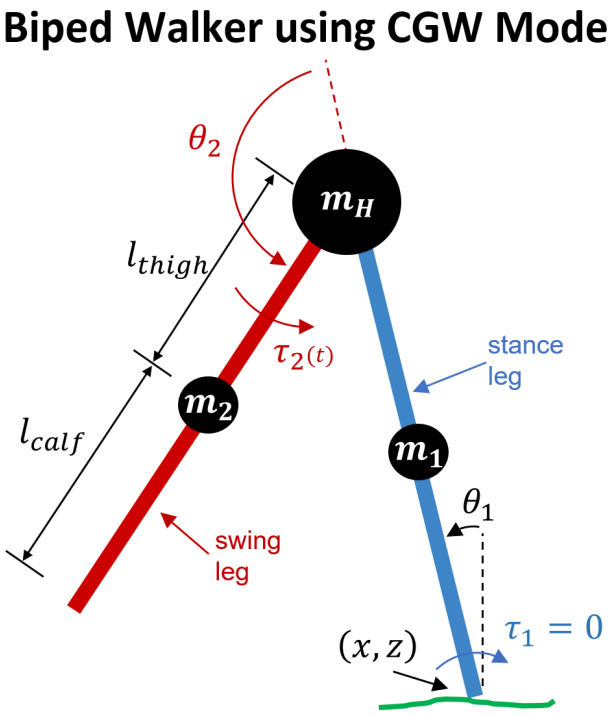

Figure 3.7: Biped Walker using the Compass Gait Walker Model. Point Masses at the hip and knees, with a pin joint at the stance foot.

dynamics in (3.20) and (3.21).

3.2.2 Compass Gait Walker (CGW) Model

The Compass Gait Walking model (CGW) is closely related to the NIP model [71], [90]. Kinematically, the CGW is the same as the NIP, however the main difference is that the CGW assumes there are masses at the knees of the system as shown in Fig. 3.7. In this model, the continuous dynamics and footstep placement depend explicitely on the motion of the swing leg. The DH parameters are given in Table 3.2.

Table 3.2: Denavit-Hartenberg Parameters and Mass Properties for Compass Gait Walker

Link	a_i	α_i	d_i	θ_i	m_i	l_{c_i}	j_i	Description
1	l_{calf} $+l_{thigh}$	0	0	θ_1	m_1 $+m_H$	$\frac{l_{calf}m_1+l_{thigh}m_H}{m_1+m_H}$	$(l_{calf}+l_{thigh}-l_{c_1})^2m_H$ $+(l_{c_1}-l_{calf})^2m_1$	stance leg
2	l_{thigh} $+l_{calf}$	0	0	θ_2	m_2	l_{thigh}	0	swing leg

The dynamics for the CGW system, in the form (3.4), are defined by

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} 2\left(c_1+c_2\right)\left(1+\cos\theta_2\right)+c_3 & c_1+\left(c_1+c_2\right)\cos(\theta_2) \\ c_1+\left(c_1+c_2\right)\cos(\theta_2) & c_1 \end{bmatrix} \tag{3.22}$$

$$\mathbf{C}\left(\mathbf{q},\dot{\mathbf{q}}\right) = \begin{bmatrix} -\left(c_1+c_2\right)\sin\theta_2\dot{\theta}_2 & -\left(c_1+c_2\right)\left(\dot{\theta}_1+\dot{\theta}_2\right) \\ \left(c_1+c_2\right)\sin\theta_2\dot{\theta}_2 & 0 \end{bmatrix} \tag{3.23}$$

$$\mathbf{g}\left(\mathbf{q}\right) = \begin{bmatrix} c_4\sin(\theta_1)+c_5\sin(\theta_1+\theta_2) \\ c_5\sin(\theta_1+\theta_2) \end{bmatrix}, \tag{3.24}$$

with constant parameters: $c_1 = m_2 l_{thigh}^2$, $c_2 = m_2 l_{calf} l_{thigh}$, $c_3 = m_2 l_{calf}^2 + \frac{\left(l_c m_1 + l_c m_H + l_t m_H\right)^2}{m_1+m_H}$, $c_4 = -g\left(m_1 l_c + \left(m_2+m_H\right)\left(l_c+l_t\right)\right)$, and $c_5 = -g m_2 l_{thigh}$. The impact map, in the form (3.12), is given by

$$\mathcal{Q}_{\mathbf{q}} = \begin{bmatrix} \theta_1+\theta_2-\pi \\ -\theta_2 \end{bmatrix}^- \tag{3.25}$$

$$\mathcal{Q}_{\dot{\mathbf{q}}^+} = \begin{bmatrix} c_6+c_7\cos\theta_2 & c_6 \\ c_6+c_8+c_9\cos\theta_2 & c_6+c_7\cos\theta_2 \end{bmatrix}^- \tag{3.26}$$

$$\mathcal{Q}_{\dot{\mathbf{q}}^-} = \begin{bmatrix} c_{10} & 0 \\ c_{10}-c_2+c_{11}\cos\theta_2 & -c_2 \end{bmatrix}^- \tag{3.27}$$

$$\mathcal{Q}_{\text{xz}} = \begin{bmatrix} x-\left(l_{thigh}+l_{calf}\right)\left(\sin\theta_1+\sin(\theta_1+\theta_2)\right) \\ z+\left(l_{thigh}+l_{calf}\right)\left(\cos\theta_1+\cos(\theta_1+\theta_2)\right) \end{bmatrix}^- \tag{3.28}$$

with constant parameters: $c_6 = \frac{m_1^2 l_{thigh}^2}{m_1+m_H}$, $c_7 = m_1 l_{thigh}\left(l_{calf}+l_{thigh}\right)$, $c_8 = \frac{m_1 m_2 l_c^2}{m_1+m_H} + \left(m_1+m_H\right)\left(l_{calf}+l_{thigh}\right)^2$, $c_9 = 2m_1 l_{thigh}\left(l_{thigh}+l_{calf}\right)$, $c_{10} = -\frac{m_1 l_{thigh}\left(m_1 l_{calf}+m_H l_{calf}+m_H l_{thigh}\right)}{m_1+m_H}$, and $c_{11} = -\left(l_{calf}+l_{thigh}\right)\left(l_{calf}\left(m_1+m_2+m_H\right)+m_H l_{thigh}\right)$.

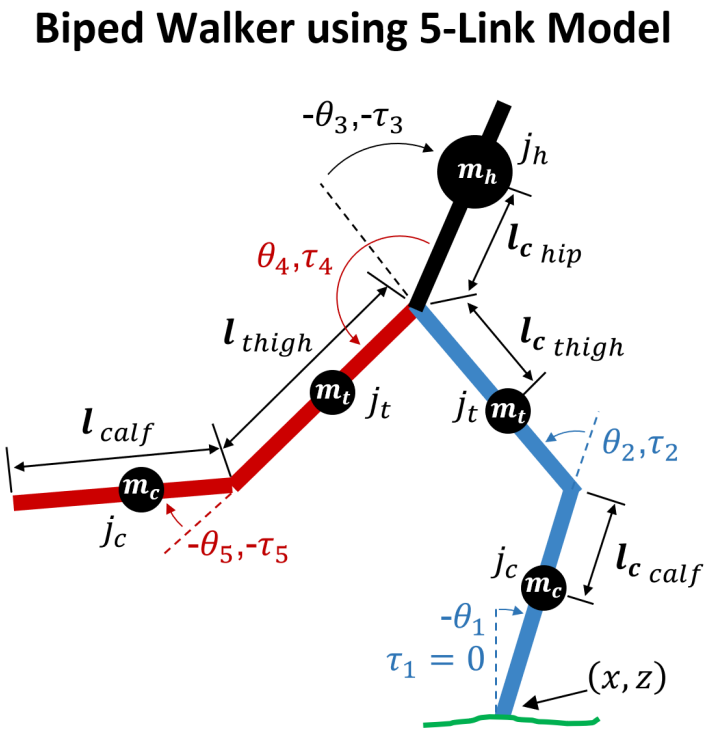

Figure 3.8: Biped Walker using the 5-Link Model. Each of the 5 links has a moment of inertia j_i about its center of mass m_i, located at length l_{c_i} from the proximal joint. Torque is available at the hips and knees, and the system has a pin joint at the stance foot.

3.2.3 5-Link Model

The final model we consider in this work is the 5-link biped. This system is representative of a humanoid as it has bending knees and a torso, as shown in Fig. 3.8. We assume that the system is symmetric DH parameters defined in Table 3.3. In this work, we create the

Table 3.3: Denavit-Hartenberg Parameters and Mass Properties for 5-Link Walker

Link	a_i	α_i	d_i	θ_i	m_i	l_{c_i}	j_i	Description
1	l_{calf}	0	0	θ_1	m_{calf}	$l_{calf} - l_{c_{calf}}$	j_{calf}	stance calf
2	l_{thigh}	0	0	θ_2	m_{thigh}	$l_{thigh} - l_{c_{thigh}}$	j_{thigh}	stance thigh
3	0	0	0	θ_3	m_{hip}	$l_{c_{hip}}$	j_{hip}	torso
4	l_{thigh}	0	0	θ_4	m_{thigh}	$l_{c_{thigh}}$	j_{thigh}	swing thigh
5	l_{calf}	0	0	θ_5	m_{calf}	$l_{c_{calf}}$	j_{calf}	swing calf

matrices $\mathbf{M}, \mathbf{C}, \mathbf{g}$ symbolically for online computation of the continuous dynamics f and the configuration matrices $\mathcal{Q}_{\mathbf{q}}, \mathcal{Q}_{\dot{\mathbf{q}}+}, \mathcal{Q}_{\dot{\mathbf{q}}-}, \mathcal{Q}_{\text{xz}}$ necessary to generate the discrete impact map M.

In this chapter, we developed three bipedal models of increasing complexity, which will be utilized in the subsequent chapters. Understanding the interaction between the continuous and discrete elements of the hybrid dynamics is essential to grasp the next chapter where we develop the unified architecture.

Chapter 4

Unified Planning for Legged Systems in Real-Time

The unified planning and control framework presented in this work draws inspiration from the hybrid systems, nonlinear analysis, and nonlinear control communities. We add control to the discrete and continuous layers of the hybrid system derived in Chap. 3. Then we project it onto a lower dimensional manifold for optimization by a trajectory optimization framework. We will show how the unified method is able to simultaneously place footholds, shape trajectories, and tune feedback controllers.

4.1 Hybrid Control

We assume a hybrid system that can handle once-per-step control parameter updates. In this work, we separate them into two categories, as shown in Fig. 4.1:

- $\mathbf{u}_{\mathrm{I}}$: discrete actions such as impulses that occur during the instantaneous state transition of contact. Examples of impulsive control include forces created by an instantaneous toe push-off.

- $\mathbf{u}_{\mathrm{K}}$: parameters that are used to control the trajectory of the continuous phase such as feedback gains or the coefficients of a polynomial representing the trajectory shape

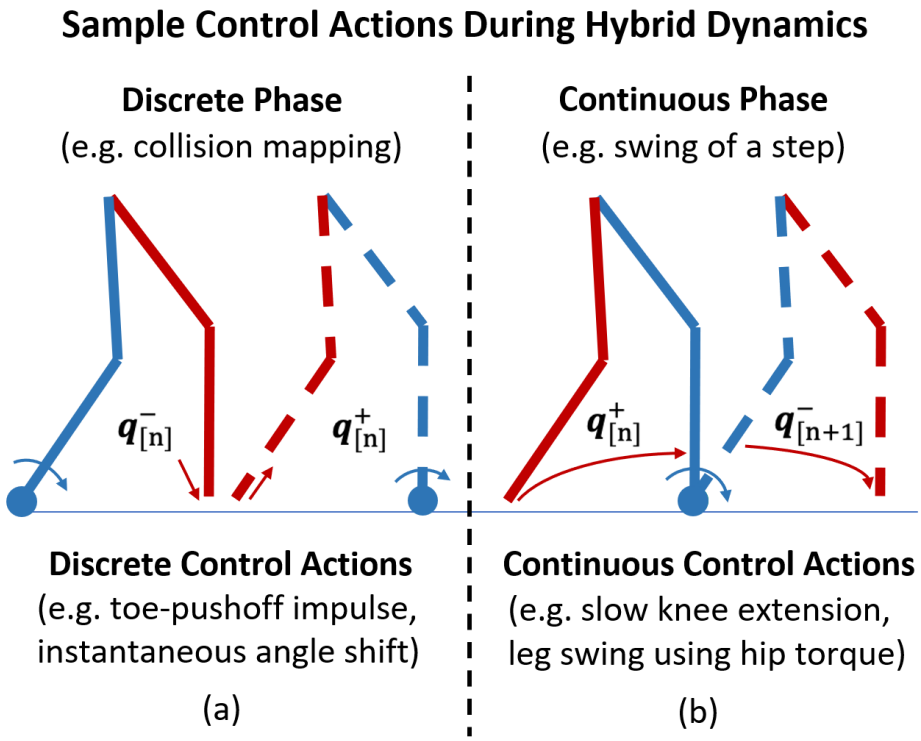

Figure 4.1: Sample Control Actions during Hybrid Dynamics. (a) The collision is mapped from some state before impact $\mathbf{q}^-$ to a post-collision state $\mathbf{q}^+$ by discrete impact function M. An instantaneous control action such as an impulse or movement of a massless limb could be present. (b) The swing phase represents continuous dynamics $\dot{\mathbf{q}}(t) = f\,(\,\mathbf{q}(\mathrm{t}), \boldsymbol{\tau}(\mathrm{t})\,)$ integrated until the time of collision $t^-_{c[n+1]}$.

We now add control $\mathbf{u}_\mathrm{I}$ to our impact map from (3.1). This transition will consist of both non-conservative energy loss due to collision with the ground and the applied impulses to generate the post-collision state

$$\mathbf{q}^+ = M(\mathbf{q}^-, \mathbf{u}_\mathrm{I}). \tag{4.1}$$

Likewise, we modify the continuous controller from (3.5) to handle once-per-step parameter tuning. Instead of using a time-varying control like $\boldsymbol{\tau}(t)$, we use time-independent parameters $\mathbf{u}_\mathrm{K}$ from the continuous controller that define $\boldsymbol{\tau}(t)$

$$\dot{\mathbf{q}} = f\,(\mathbf{q}(\mathrm{t}), \mathbf{u}_\mathrm{K})\,. \tag{4.2}$$

One such example of time-independent parameters $\mathbf{u}_\mathrm{K}$ are the PD gains for joint control

of a single step. For example, if an outer loop changed the target joint angles, it would be ideal if the outer loop could also tune the inner-loop gains to optimize the inner response.

Control $\mathbf{u}_{[\mathrm{n}]}$ is the vector of m once-per-step control parameters for a given step n

$$\mathbf{u}_{[\mathrm{n}]} = \begin{bmatrix} \mathbf{u}_{\mathrm{I}} \\ \\ \mathbf{u}_{\mathrm{K}} \end{bmatrix}. \tag{4.3}$$

Now, using (4.1), (4.2), and (4.3), the controlled hybrid map (variant of (3.3)) is now

$$\mathbf{q}^{-}_{[\mathrm{n+1}]} = M\left(\mathbf{q}^{-}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}\right) + \int_{t^{+}_{c[\mathrm{n}]}}^{t^{-}_{c[\mathrm{n+1}]}} f\big(\mathbf{q}(\mathrm{t}), \mathbf{u}_{[\mathrm{n}]}\big)\,\mathrm{dt}, \tag{4.4}$$

where $t^{+}_{c[\mathrm{n}]}$ represents the time of collision n marked by post-collision state $\mathbf{q}^{+}_{[\mathrm{n}]}$; and $t^{-}_{c[\mathrm{n+1}]}$ represents the time of collision n+1 marked with pre-collision state $\mathbf{q}^{-}_{[\mathrm{n+1}]}$. It is important to understand that the control $\mathbf{u}_{[\mathrm{n}]}$ is not time dependent, but rather step n dependent, an important distinction that necessarily differentiates our unified framework from other works.

4.1.1 Projected Mapping

For a walking system, (4.4) will repeat itself at every successful step. In dynamical systems, a return map, or Poincaré map [91], is a useful tool for analyzing the behavior of such systems. We will demonstrate how to create such a mapping, and we will explore its particular usefulness for a walking system in Sect. 4.1.3.

We mark the intersection of contact with the ground as a section that is destined to repeat through the same cycle. This surface of section *s.o.s.* [92] is a lower-dimensional subspace $\mathbf{x}$ that captures the entire properties of the orbit itself. For instance, consider a 2D pendulum with state $\mathbf{q} = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^{\mathrm{T}}$ that could be sliced at $\theta = 0$ whenever $\dot{\theta} > 0$. When the state space is projected onto this slice, θ disappears and only $\dot{\theta}$ is relevant. Additionally, only $\dot{\theta}$ and the definition of the section ($\theta = 0$) are necessary to reconstruct the full state space. Analytically, we define this projection as D, with inverse D^{-1}, that

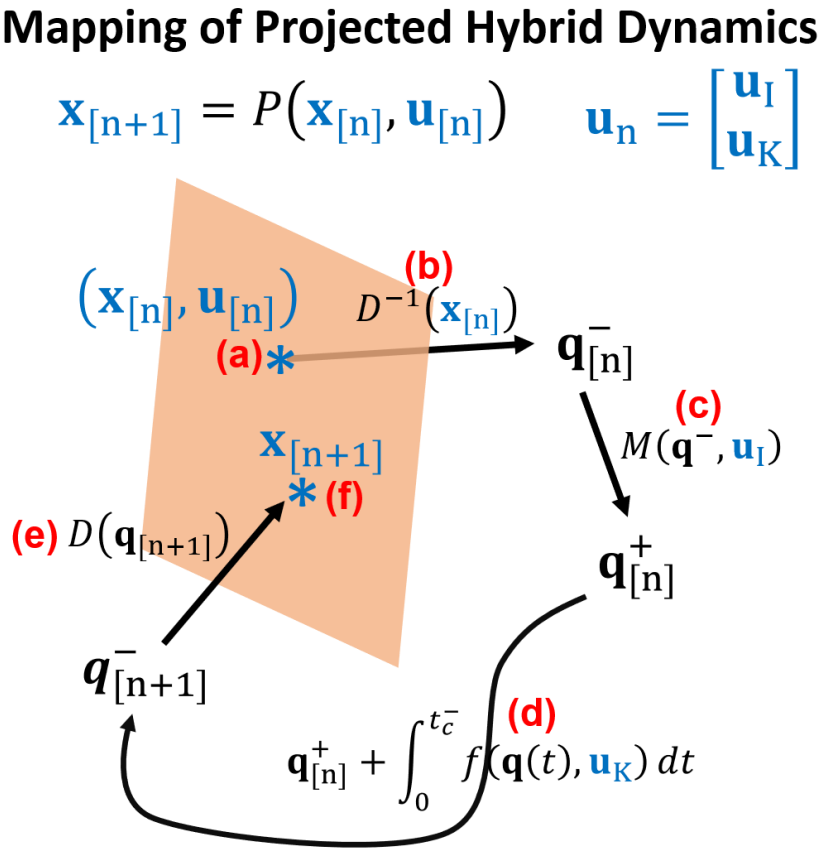

Figure 4.2: Mapping of Projected Hybrid Dynamics. This projection slices the surface of section *s.o.s.* (a) immediately before impact: (b) dimensionality reduction D, (c) impact mapping M of contact, (d) continuous dynamics f of the swing phase, and (e) inverse dimensionality D^{-1} to arrive back at the mapping at a new fixed state $\mathbf{x}_{[n+1]}$ (f).

we assume is defined uniformly

$$\begin{aligned} \mathbf{x} &= D(\mathbf{q}) \\ \mathbf{q} &= D^{-1}(\mathbf{x}). \end{aligned} \tag{4.5}$$

We define the mapping P from one *s.o.s.* $\mathbf{x}_{[n]}$ to the next $\mathbf{x}_{[n+1]}$ as

$$\mathbf{x}_{[n+1]} = \mathrm{P}\left(\mathbf{x}_{[n]}, \mathbf{u}_{[n]}\right). \tag{4.6}$$

We will utilize Fig. 4.2 to explain how the mapping (4.6) fits the hybrid model. This work defines the return map at the time immediately before contact t_c^- for the n^{th} step.

a) The reduced state $\mathbf{x}_{[\mathrm{n}]}$ and once-per-step control actions $\mathbf{u}_{[\mathrm{n}]}$ are defined on the surface.

b) The mapping completes a dimensionality increase D^{-1} to reinstate the full system state space $\mathbf{q}^{-}_{[\mathrm{n}]}$ at time $t^{-}_{c_n}$. Additionally, the control action $\mathbf{u}_{[\mathrm{n}]}$ is deconstructed into its discrete impulses $\mathbf{u}_{\mathrm{I}_{[\mathrm{n}]}}$ and continuous parameters $\mathbf{u}_{\mathrm{K}_{[\mathrm{n}]}}$ as per (4.3).

c) At contact, discrete mapping M represents the combination of the inelastic collision with the ground and discrete control actions $\mathbf{u}_{\mathrm{I}_{[\mathrm{n}]}}$ to generate the post-collision state $\mathbf{q}^{+}_{[\mathrm{n}]}$.

d) From post-impact, the dynamics f of the swing phase are integrated until next contact with the ground at $\mathbf{q}^{-}_{[\mathrm{n+1}]}$. The swing phase utilizes the continuous control parameters $\mathbf{u}_{\mathrm{K}_{[\mathrm{n}]}}$ to fully define its feedback controller for the entirety of the n^{th} step.

e) The system is then reduced back onto its surface of section by the discrete projection function D.

f) We end at $\mathbf{x}_{[\mathrm{n+1}]}$, back on the *s.o.s.*

We describe this projection mapping analytically by

$$\begin{aligned} P\left(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}\right) =& D\left[\mathbf{q}\left(\mathrm{t}^{-}_{\mathrm{c[n+1]}}, \mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}\right)\right] \\ =& D\left[\mathbf{q}^{+}_{[\mathrm{n}]} + \int_{\mathrm{t}^{+}_{\mathrm{c[n]}}}^{\mathrm{t}^{-}_{\mathrm{c[n+1]}}} f\big(\mathbf{q}(\mathrm{t}), \mathbf{u}_{[\mathrm{n}]}\big)\,\mathrm{dt}\right] \\ =& D\left[M\left(D^{-1}\left(\mathbf{x}_{[\mathrm{n}]}\right), \mathbf{u}_{[\mathrm{n}]}\right) + \int_{t^{+}_{\mathrm{c[n]}}}^{t^{-}_{\mathrm{c[n+1]}}} f\left(\mathbf{q}\left(\mathrm{t}, \mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}\right), \mathbf{u}_{[\mathrm{n}]}\right) dt\right]. \end{aligned} \tag{4.7}$$

We must note that (4.7) is not a closed-form solution of the hybrid dynamics, but rather involves an integration until next contact, determined by

$$\phi\left(q\left(t^{-}_{c[\mathrm{n+1}]}, \mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}\right)\right) = 0, \tag{4.8}$$

where ϕ is a constraint denoting collision of the foot with the terrain surface. For a walking system, (4.7) is equivalent to propagating the dynamics forward until the swing

foot comes in contact with the terrain map (4.8) at $t^-_{c[\mathrm{n}+1]}$.

4.1.2 Linearization of the Projected Dynamics

In Sect. 4.1, we derived the projected dynamics P (4.6), which where shown to represent a discrete slice of a trajectory known as a *s.o.s.* In the following section, we will then derive a discrete-time feedback controller to "regulate the system on the *s.o.s.*."

We first compute the linearization of (4.6) by computing a first order Taylor Series expansion about fixed point $\left(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}\right)$

$$\begin{aligned}
\mathbf{x}_{[\mathrm{n}+1]} &= \mathbf{x}^*_{[\mathrm{n}+1]} + \mathbf{A}^*_{[\mathrm{n}]}\left(\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^*_{[\mathrm{n}]}\right) + \mathbf{B}^*_{[\mathrm{n}]}\left(\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^*_{[\mathrm{n}]}\right) \\
\bar{\mathbf{x}}_{[\mathrm{n}+1]} &= \mathbf{A}^*_{[\mathrm{n}]}\bar{\mathbf{x}}_{[\mathrm{n}]} + \mathbf{B}^*_{[\mathrm{n}]}\bar{\mathbf{u}}_{[\mathrm{n}]}
\end{aligned} \tag{4.9}$$

where

$$\mathbf{A}^*_{[\mathrm{n}]} = \frac{\partial P\left(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}\right)}{\partial \mathbf{x}_{[\mathrm{n}]}}, \;\; \mathbf{B}^*_{[\mathrm{n}]} = \frac{\partial \mathrm{P}\left(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}\right)}{\partial \mathbf{u}_{[\mathrm{n}]}}.$$

While some passive biped walking systems (e.g. rimless wheel, CGW) can demonstrate an unforced natural response that is stable [75] when moving downhill, the majority of systems require active control. Introducing the idea of active control on a finite-horizon *s.o.s.* to optimize walking for a legged system is a novel contribution in this work.

Computing the linearized dynamics as defined on the *s.o.s.* ($\mathbf{A}^*_{[\mathrm{n}]}$,$\mathbf{B}^*_{[\mathrm{n}]}$) consist of deriving the full linearization of (4.7), which can be given as

$$\frac{\partial P}{\partial \mathbf{x}_{[\mathrm{n}]}} = \frac{\partial D}{\partial \mathbf{q}}\frac{\partial M}{\partial \mathbf{q}}\left[f(q(t))\frac{\partial t^-_{c[\mathrm{n}+1]}}{\partial \mathbf{x}_{[\mathrm{n}]}} + \frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{x}_{[\mathrm{n}]}}\right]\Bigg|_{\iota} \tag{4.10}$$

$$\frac{\partial P}{\partial \mathbf{u}_{[\mathrm{n}]}} = \frac{\partial D}{\partial \mathbf{q}}\left[\frac{\partial M}{\partial \mathbf{q}}\left[f(q(t))\frac{\partial t^-_{c[\mathrm{n}+1]}}{\partial \mathbf{u}_{[\mathrm{n}]}} + \frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}}\right] + \frac{\partial M}{\partial \mathbf{u}_{[\mathrm{n}]}}\right]\Bigg|_{\iota} \tag{4.11}$$

and evaluated at $t = t^-_{c[\mathrm{n}+1]}$, time of next ϕ contact. Full derivation, method for solving, and definition of "ι" are shown in App. B.

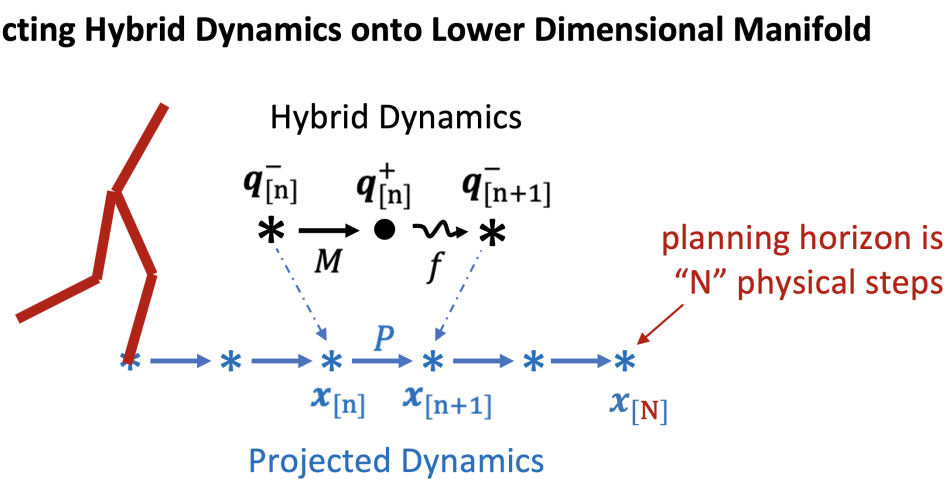

Figure 4.3: Projection of Hybrid Dynamics onto a Lower Dimensional Manifold. Hybrid dynamics of a periodic orbit fully represented as two points of discrete dynamics where $\mathbf{x}_{[n+1]} = \mathrm{P}(\mathbf{x}_{[n]}, \mathbf{u}_{[n]})$. Control $\mathbf{u}_{[n]}$ represents all parameters that can be changed once-per-step, such as instantaneous actions and feedback gains.

4.1.3 Nonlinear Analysis of the Projected Dynamics

The hybrid dynamics are projected onto a lower dimensional surface P, as shown in Fig 4.3. This section will demonstrate how to simultaneously adjust states and the once-per-step controller on the *s.o.s.*, which will be shown to have some added benefit for a walking system.

This section is used to construct a controller to stabilize the system of the form

$$\bar{\mathbf{x}}_{[n+1]} = \left(\mathbf{A}^*_{[n]} - \mathbf{B}^*_{[n]}\mathbf{K}^{\mathrm{fb}}_{[n]}\right)\bar{\mathbf{x}}_{[n]}. \tag{4.12}$$

We will show how the iLQR update adjusts the old feedback controller $\mathbf{K}^{\mathrm{fb}}_{[n]old}$ to a new controller $\mathbf{K}^{\mathrm{fb}}_{[n]new}$ (denoted by the subscript *old* and *new*, respectively), thus changing the linearization fixed point from $\left(\mathbf{x}^*_{[n]}, \mathbf{u}^*_{[n]old}\right)$ to $\left(\mathbf{x}^*_{[n]}, \mathbf{u}^*_{[n]new}\right)$. Classically, a first-order iLQR algorithm would update the controller with $\mathbf{u}^*_{[n]new} = \mathbf{u}^*_{[n]old} + \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}$. Such an

update to (4.12) looks like

$$\begin{aligned}
\bar{\mathbf{x}}_{[n+1]} &= \mathbf{A}^*_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}^*_{[n]}\left(\mathbf{u}_{[n]} - \mathbf{u}^*_{[n]new}\right) && \text{using } \bar{\mathbf{u}}_{[n]} = \mathbf{u}_{[n]} - \bar{\mathbf{u}}^*_{[n]} \\
&= \mathbf{A}^*_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}^*_{[n]}\left(\mathbf{u}_{[n]} - \mathbf{u}^*_{[n]old} - \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}\right) && \text{using } \mathbf{u}^*_{[n]new} = \mathbf{u}^*_{[n]old} + \mathbf{K}^{\text{fb-inc}}_{[n]} \\
&= \mathbf{A}^*_{[n]}\bar{\mathbf{x}}_{[n]} - \mathbf{B}^*_{[n]}\left(\left(\mathbf{K}^{\text{fb}}_{[n]old} + \mathbf{K}^{\text{fb-inc}}_{[n]}\right)\bar{\mathbf{x}}_{[n]}\right) && \text{using } \mathbf{u}_{[n]} - \mathbf{u}^*_{[n]old} = -\mathbf{K}^{\text{fb}}_{[n]old}\bar{\mathbf{x}}_{[n]} \\
&= \left[\mathbf{A}^*_{[n]} - \mathbf{B}^*_{[n]}\left(\mathbf{K}^{\text{fb}}_{[n]old} + \mathbf{K}^{\text{fb-inc}}_{[n]}\right)\right]\bar{\mathbf{x}}_{[n]} \\
&= \left[\mathbf{A}^*_{[n]} - \mathbf{B}^*_{[n]}\mathbf{K}^{\text{fb}}_{[n]new}\right]\bar{\mathbf{x}}_{[n]} && \text{using } \mathbf{K}^{\text{fb}}_{[n]new} = \mathbf{K}^{\text{fb}}_{[n]old} + \mathbf{K}^{\text{fb-inc}}_{[n]}.
\end{aligned} \tag{4.13}$$

(4.13) should make it clear that the first-order iLQR algorithm is updating, not replacing, the old feedback gains with a feedback increment to optimally stabilize the system about the original fixed point $\mathbf{x}^*_{[n+1]}$.

We demonstrated the first-order incremental update to the gain matrix $\mathbf{K}$ in (4.13), but our system is built using a second order version of finite-horizon SLQ Regulator as derived in App. A. One key difference for our 2nd order derivation also adds a feedforward increment $\mathbf{u}^{\text{ff-inc}}$ such that

$$\mathbf{u}^*_{[n]\text{new}} = \mathbf{u}^*_{[n]old} + \mathbf{u}^{\text{ff-inc}}_{[n]} + \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}. \tag{4.14}$$

When we add (4.14) to our system, an additional term $\mathbf{B}^*_{[n]}\mathbf{u}^{\text{ff-inc}}_{[n]}$ shows up

$$\begin{aligned}
\bar{\mathbf{x}}_{[n+1]} &= \mathbf{A}^*_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}^*_{[n]}\left(\mathbf{u}_{[n]} - \mathbf{u}^*_{[n]new}\right) \\
&= \mathbf{A}^*_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}^*_{[n]}\left(\mathbf{u}_{[n]} - \mathbf{u}^*_{[n]old} - \mathbf{u}^{\text{ff-inc}}_{[n]} - \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}\right) && \text{using (4.14)} \\
&= \left[\mathbf{A}^*_{[n]} - \mathbf{B}^*_{[n]}\left(\mathbf{K}^{\text{fb}}_{[n]old} + \mathbf{K}^{\text{fb-inc}}_{[n]}\right)\right]\bar{\mathbf{x}}_{[n]} - \mathbf{B}^*_{[n]}\mathbf{u}^{\text{ff-inc}}_{[n]} && \text{using (4.13).}
\end{aligned} \tag{4.15}$$

The addition of this new term $-\mathbf{B}^*_{[n]}\mathbf{u}^{\text{ff-inc}}_{[n]}$ is important to the context of walking

because it moves the fixed point. Assuming $\bar{\mathbf{x}}_{[\mathrm{n}]} = \mathbf{0}$, (4.15) becomes

$$\begin{aligned} \bar{\mathbf{x}}^*_{[n+1]old} &= -\mathbf{B}^*_{[\mathrm{n}]}\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]} \\ \mathbf{x}_{[\mathrm{n+1}]} - \mathbf{x}^*_{[\mathrm{n+1}]old} &= -\mathbf{B}^*_{[\mathrm{n}]}\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]} \\ \mathbf{x}_{[\mathrm{n+1}]} &= \mathbf{x}^*_{[\mathrm{n+1}]old} - \mathbf{B}^*_{[\mathrm{n}]}\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]}. \end{aligned} \tag{4.16}$$

By establishing that the fixed point is moving in (4.16), we can define the state space adjustment $\mathbf{x}^{*-\text{inc}}_{[\mathrm{n+1}]}$ caused by the feedforward increment as

$$\mathbf{x}^{*-\text{inc}}_{[\mathrm{n+1}]} = -\mathbf{B}^*_{[\mathrm{n}]}\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]} \tag{4.17}$$

and arrive at

$$\begin{aligned} \mathbf{x}^*_{[\mathrm{n+1}]new} &= \mathbf{x}^*_{[\mathrm{n+1}]old} - \mathbf{B}^*_{[\mathrm{n}]}\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]} \\ &= \mathbf{x}^*_{[\mathrm{n+1}]old} + \mathbf{x}^{*-\text{inc}}_{[\mathrm{n+1}]}. \end{aligned} \tag{4.18}$$

By joining (4.18) and the relationship $\mathbf{K}^{\text{fb}}_{[\mathrm{n}]new} = \mathbf{K}^{\text{fb}}_{[\mathrm{n}]old} + \mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]}$ into (4.15), we arrive at an expression with both a feedforward and feedback term around the new, SLQ-modified fixed point $\left(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]new}\right)$

$$\bar{\mathbf{x}}_{[\mathrm{n+1}]new} = \left(\mathbf{A}^*_{[\mathrm{n}]} - \mathbf{B}^*_{[\mathrm{n}]}\mathbf{K}^{\text{fb}}_{[\mathrm{n}]new}\right)\bar{\mathbf{x}}_{[\mathrm{n}]}. \tag{4.19}$$

(4.16)-(4.19) provide some very important insight into what the feedforward control increment creates an adjustment to the target fixed point $\mathbf{x}^*_{[\mathrm{n+1}]old}$ on the *s.o.s.* In the context of the walking robot, $\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]}$ is directly altering where the robot steps while $\mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]}$ provides the additional stability required in order to stabilize around the new target step location $\mathbf{x}^*_{[\mathrm{n+1}]new}$ as shown in Fig. 4.4. Additionally, since each controller is tuned to the n^{th} step, we iterate forward and backwards until the control increments disappear below a set tolerance and $\mathbf{x}^*_{[\mathrm{n+1}]new} = \mathbf{x}^*_{[\mathrm{n+1}]old}$ and the planned foothold stops moving.

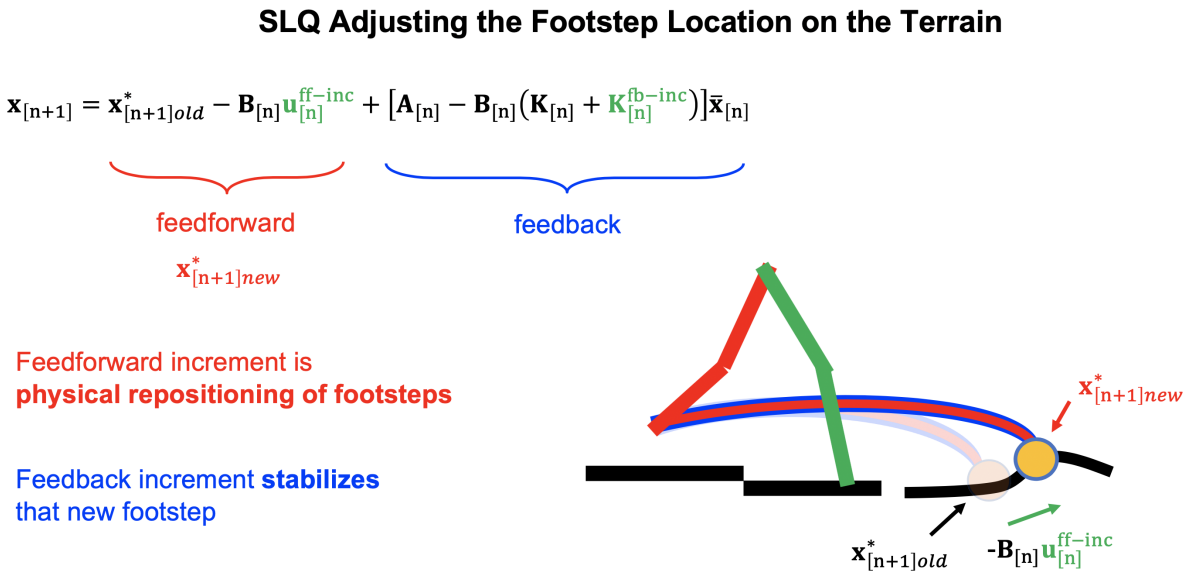

Figure 4.4: SLQ Adjusting the Footstep Location on the Terrain. The feedforward control increment $\mathbf{u}^{\text{ff}-\text{inc}}_{[n]}$ derived from SLQ adjusts the target footstep location on the terrain, re-positioning the target from $\mathbf{x}^*_{[n+1]old}$ to $\mathbf{x}^*_{[n+1]new}$. The feedback increment $\mathbf{K}^{\text{fb}-\text{inc}}_{[n]}$ helps stabilize the motion around this newly-created trajectory.

Fig. 4.5 depicts a simple hybrid system being mapped within an MPC framework. In the outer loop, footsteps are optimized using the second order SLQ method discussed above. Inside the discrete step, the entire process from (4.7) occurs. On the SLQ forward pass, the entire hybrid dynamics are simulated forward. During the SLQ backward pass, the control $\mathbf{u}^*_{[n]}$ and targets for state $\mathbf{x}^*_{[n+1]}$ are updated per (4.14)-(4.18). To account for discretization error, the feedforward element is iteratively adjusted using a line search to ensure selection of the optimal solution. The entire process repeats until convergence of $\left\|\mathbf{u}^{\text{ff-inc}}_{[n]}\right\|$ $\left(\text{and thus } \left\|\mathbf{x}^{*-\text{inc}}_{[n+1]}\right\|\right)$ below an acceptable tolerance.

4.2 Nonlinear Planning and Control in Unstructured Terrain

We intend to generate continuous maps that approximate the local terrain. Then we assign terrain costs based on those maps. Subsequently, the terrain costs will be incorporated into the planning framework to help simultaneously plan footsteps and derive

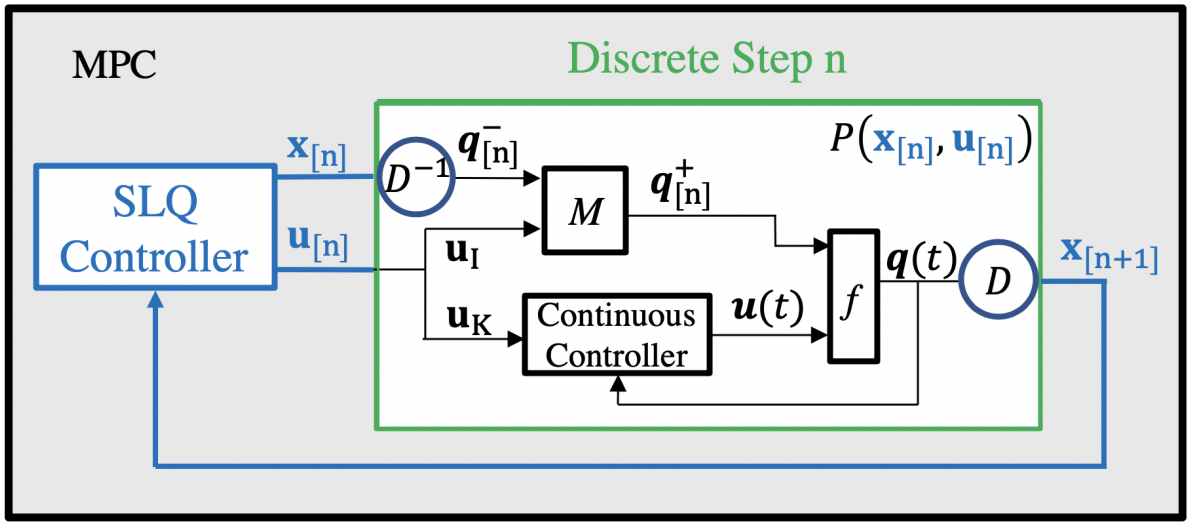

Figure 4.5: Hybrid Dynamics with discrete controllable parameters $\mathbf{u}_{[n]}$ tuned by an SLQ controller operating under an MPC architecture. Discrete actions $\mathbf{u}_I$ represent impulses or other such instantaneous actions occurring at impact map M. Parameters $\mathbf{u}_K$ are used to control the continuous phase f, initiated after impact mapping. Projection mapping D and its respective inverse D^{-1} encountered leaving and entering the mapping. The entire discrete step n can be denoted by discrete projection mapping P. While N steps are planned by SLQ controller, only a single step is shown for clarity.

controlled motion plans.

To produce a set of appropriate footholds, a footstep selection planner first needs terrain cost information regarding desirability of the nominal stepping locations $\mathbf{X}^*$ (as shown in Fig. 1.1). A terrain cost point cloud could be generated for a landscape from some terrain-perception based on desirability of footstep locations. Local approximations of continuous cost functions should created using subsets of this data. Data is selected in the region of footstep impact and footstep adjustments will be made based on the local gradient as shown in Fig. 4.6. The creation of a terrain cost map based on environmental information gathered is dependent upon the robot and task, which is beyond the scope of this work. However, We assume that the terrain perception model operates fast enough in generating terrain cost maps so as to not create a planning bottleneck for the system.

We fit a polynomial $p_{[n]}(\mathbf{x})$ to a point cloud in the local vicinity of $\mathbf{x}^*_{[n]}$. Fig. 4.6 shows second order polynomials generated using data at $\pm 20\%$ of the nominal step size, but size of the region will be dependent upon the task and quality of the data. Polynomials must

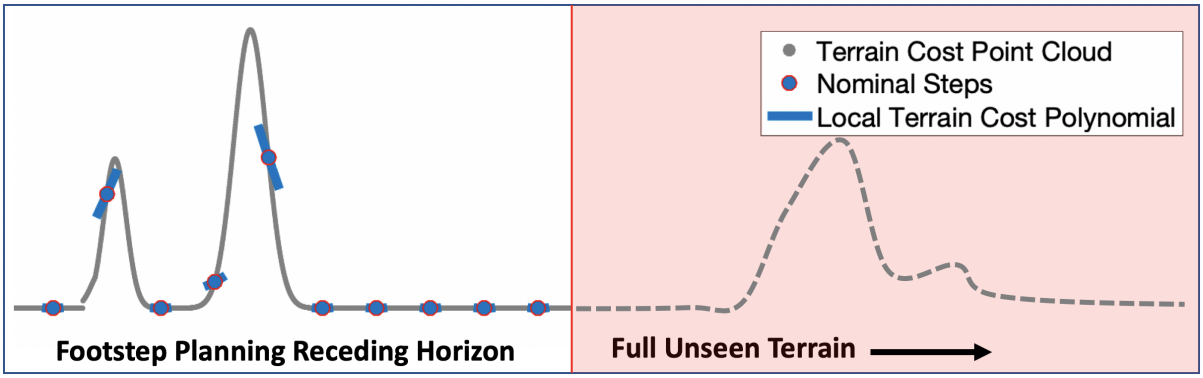

Figure 4.6: Local 2^{nd} order terrain cost polynomials overlaid onto 1D terrain cost point cloud shown with a receding horizon.

be at least second order such that the terrain cost does not disappear at the Hessian of the SLQ value function (see App. A).

4.2.1 Sequential Linear Quadratic Regulator Between Poincare Maps

Next, we give more details upon how we combine both the discrete linearization of the hybrid dynamics (4.9) and the terrain cost into a single SLQ optimization that will derive an optimal feedback controller, design the motion in between contacts, while simultaneously planning feasible foothold locations over the terrain map. In (4.9), the system is linearized about an initial sequence of trajectories and we showed how to implement control increments in Sect. 4.1.3. This section will show how to solve for new control increments using discrete SLQ.

Discrete SLQ addresses the problem

$$\min_{\mathbf{u}(\cdot)} \left\{ \bar{\Phi}(\mathbf{x}_{[\mathrm{N}]}) + \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \bar{\mathrm{L}}_{[\mathrm{n}]} \left(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]} \right) \right\}$$

$$\text{subject to: } \mathbf{x}_{[\mathrm{n}+1]} = \mathrm{P}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) \qquad \mathbf{x}_{[0]} = \mathbf{x}_0, \tag{4.20}$$

$$\Phi(\mathbf{x}_{[\mathrm{N}]}) = \frac{1}{2} \mathbf{x}_{[\mathrm{N}]}^{\mathrm{T}} \mathbf{Q}_{[\mathrm{N}]} \mathbf{x}_{[\mathrm{N}]} + \mathbf{x}_{[\mathrm{N}]}^{\mathrm{T}} \mathbf{q}_{[\mathrm{N}]} + \mathrm{q}_{[\mathrm{N}]} \tag{4.21}$$

$$\mathrm{L}_{[\mathrm{n}]}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_{[\mathrm{n}]} \\ \mathbf{u}_{[\mathrm{n}]} \end{bmatrix}^T \begin{bmatrix} \mathbf{Q}_{[\mathrm{n}]} & \mathbf{P}_{[\mathrm{n}]}^{\mathrm{T}} \\ \mathbf{P}_{[\mathrm{n}]} & \mathbf{R}_{[\mathrm{n}]} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{[\mathrm{n}]} \\ \mathbf{u}_{[\mathrm{n}]} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{[\mathrm{n}]} \\ \mathbf{u}_{[\mathrm{n}]} \end{bmatrix}^T \begin{bmatrix} \mathbf{q}_{[\mathrm{n}]} \\ \mathbf{r}_{[\mathrm{n}]} \end{bmatrix} + \begin{bmatrix} q_{[\mathrm{n}]} \\ r_{[\mathrm{n}]} \end{bmatrix} \tag{4.22}$$

where $\mathrm{P}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})$ represents the Poincare map defined in (4.7). A quadratic terrain cost term is added to the original final and running costs of (4.20):

$$\bar{\Phi}_{[\mathrm{N}]}(\mathbf{x}_{[\mathrm{N}]}) = \Phi_{[\mathrm{N}]}(\mathbf{x}_{[\mathrm{N}]}) + p_{[\mathrm{N}]}(\mathbf{x}_{[\mathrm{N}]}) \tag{4.23}$$

$$\bar{\mathrm{L}}_{[\mathrm{n}]}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) = \mathrm{L}_{[\mathrm{n}]}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) + p_{[\mathrm{n}]}(\mathbf{x}_{[\mathrm{n}]}) \tag{4.24}$$

If the terrain polynomial was less than quadratic, the terrain cost would disappear in the second order derivatives of the cost function, which are found within of the control updates, shown below. The main idea that underlies SLQ is to minimize a running cost-to-go function [45], [93]. The SLQ derivation we employ in this work is shown in App. A, but are summarized here:

$$\mathbf{g}_{[\mathrm{n}]} = \left(\bar{\mathbf{L}}^*_u + \mathbf{s}_{[\mathrm{n+1}]}\mathbf{B}_{[\mathrm{n}]}\right)^T \tag{4.25}$$

$$\mathbf{G}_{[\mathrm{n}]} = \left(\bar{\mathbf{L}}^*_{xu} + \mathbf{A}^{\mathrm{T}}_{[\mathrm{n}]}\mathbf{S}_{[\mathrm{n+1}]}\mathbf{B}_{[\mathrm{n}]}\right)^T \tag{4.26}$$

$$\mathbf{H}_{[\mathrm{n}]} = \left(\bar{\mathbf{L}}^*_{uu} + \mathbf{B}^{\mathrm{T}}_{[\mathrm{n}]}\mathbf{S}_{[\mathrm{n+1}]}\mathbf{B}_{[\mathrm{n}]}\right)^T \tag{4.27}$$

$$\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]} = -\mathbf{H}^{-1}_{[\mathrm{n}]}\mathbf{g}_{[\mathrm{n}]} \tag{4.28}$$

$$\mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]} = -\mathbf{H}^{-1}_{[\mathrm{n}]}\mathbf{G}_{[\mathrm{n}]} \tag{4.29}$$

$$\mathbf{S}_{[\mathrm{n}]} = \bar{\mathbf{L}}^*_{xx} + \mathbf{A}^{\mathrm{T}}_{[\mathrm{n}]}\mathbf{S}_{[\mathrm{n+1}]}\mathbf{A}_{[\mathrm{n}]} + \left(\mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]}\right)^{\mathrm{T}}\mathbf{G}_{[\mathrm{n}]} + \left(\left(\mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]}\right)^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}_{[\mathrm{n}]} + \mathbf{G}^{\mathrm{T}}_{[\mathrm{n}]}\right)\mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]} \tag{4.30}$$

$$\mathbf{s}_{[\mathrm{n}]} = \bar{\mathbf{L}}^*_x + \mathbf{s}_{[\mathrm{n+1}]}\mathbf{A}_{[\mathrm{n}]} + \left(\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]}\right)^{\mathrm{T}}\mathbf{G}_{[\mathrm{n}]} + \left(\left(\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]}\right)^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}_{[\mathrm{n}]} + \mathbf{g}^{\mathrm{T}}_{[\mathrm{n}]}\right)\mathbf{K}^{\text{ff-inc}}_{[\mathrm{n}]}. \tag{4.31}$$

Derivatives and seconds derivatives in (4.25)-(4.30) are addressed with subscripts in a manner as follows: $\bar{\mathbf{\Phi}}^*_u$ and $\bar{\mathbf{L}}^*_{xx}$ represent the control derivative of the endpoint cost and state Hessian of the running cost, respectively, evaluated around a linearization of the nominal trajectory.

The SLQ algorithm is summarized as follows:

1. **Nominal Seed**: Initiate with a feasible nominal seed $\mathbf{U}^* = \begin{bmatrix}\mathbf{u}^*_{[0]} & \dots & \mathbf{u}^*_{[\mathrm{N}-1]}\end{bmatrix}$ from initial state $\mathbf{x}_0$ using (4.7) to derive an initial nominal trajectory $\mathbf{X}^* = \begin{bmatrix}\mathbf{x}^*_{[0]} & \dots & \mathbf{x}^*_{[\mathrm{n}]}\end{bmatrix}$.

2. **Gradient of Seed**: Compute the derivatives $\mathbf{A}^*_{[\mathrm{n}]}, \mathbf{B}^*_{[\mathrm{n}]}$ of the linearization around the fixed point set $(\mathbf{X}^*, \mathbf{U}^*)$ as per (4.9)-(4.11).

3. **Nominal Cost**: Compute initial cost J per (4.21)-(4.22) for $(\mathbf{X}^*, \mathbf{U}^*)$.

4. **Backwards Riccati**: Solve backwards for the feedforward $\mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]}$ and feedback $\mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]}$ control increments using (4.28)-(4.29). In this step (4.31)-(4.31) are stepped backwards from N to 1. The Ricatti-scheme is initialized with $\mathbf{S}_{[\mathrm{N}]} = \bar{\mathbf{\Phi}}^*_{xx}$, $\mathbf{s}_{[\mathrm{N}]} = \bar{\mathbf{\Phi}}^*_{x}$. (4.25)-(4.27) are intermediate solutions.

5. **Line Search**: Using feedback control increment policy (4.14) that includes coefficient α to vary the amount of feedforward control is added to the system. This process accounts for discrepancies within the linearization and the fact that we are adjusting both $\mathbf{x}^*_{[\mathrm{n+1}]}$ and $\mathbf{u}^*_{[\mathrm{n}]}$ simultaneously.
   Policy Update: $\mathbf{u}^*_{[\mathrm{n}]new} = \mathbf{u}^*_{[\mathrm{n}]old} + \alpha \mathbf{u}^{\text{ff-inc}}_{[\mathrm{n}]} + \mathbf{K}^{\text{fb-inc}}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]}$
   Forward simulate (4.7): With full control increment ($\alpha = 1$). Lower α towards $\alpha = 0$ and repeat until a minimum cost J_α is reached. Note: at $\alpha = 0$, $\bar{\mathbf{x}}^*_{[\mathrm{n+1}]} = \bar{\mathbf{x}}^*_{[\mathrm{n}]} = \mathbf{0}$ and $\mathbf{x}^{*\text{-inc}}_{[\mathrm{n}]} = \mathbf{0}$.

6. Select new policy at $\min(J_\alpha)$. $\mathbf{U}^* \leftarrow \mathbf{U}^*_{\min(\mathrm{J}_\alpha)}$

7. Repeat steps (1-5) until $\left|\left|\mathbf{u}^{[\text{ff-inc}]}_{[\mathrm{n}]}\right|\right| < tol$

This is the planning framework referenced within Fig. 1.1. Steps (1-6) must occur once-per-step for a legged system for an N step horizon. For our legged system, discrete point "n" represents a footstep and the entire projected hybrid step (4.7) is linearized per (4.9). Thus, $N = 10$ represents a 10-step look-ahead optimization. The size of the horizon can be driven by either computational or vision limits. In Chap. 7, we will discuss the time it takes to complete this process on hardware. Note, our algorithm gains extra speed (discussed in Chap. 7) by combining steps (1-2) into a single analytical solution and integrating both at once.

For legged locomotion, we found that heavy weighting of the final cost $\Phi_{[\mathrm{N}]}$ (relative to intermediate costs $\mathrm{L}_{[\mathrm{n}]}$) is very important, allowing dynamic motion of the intermediate

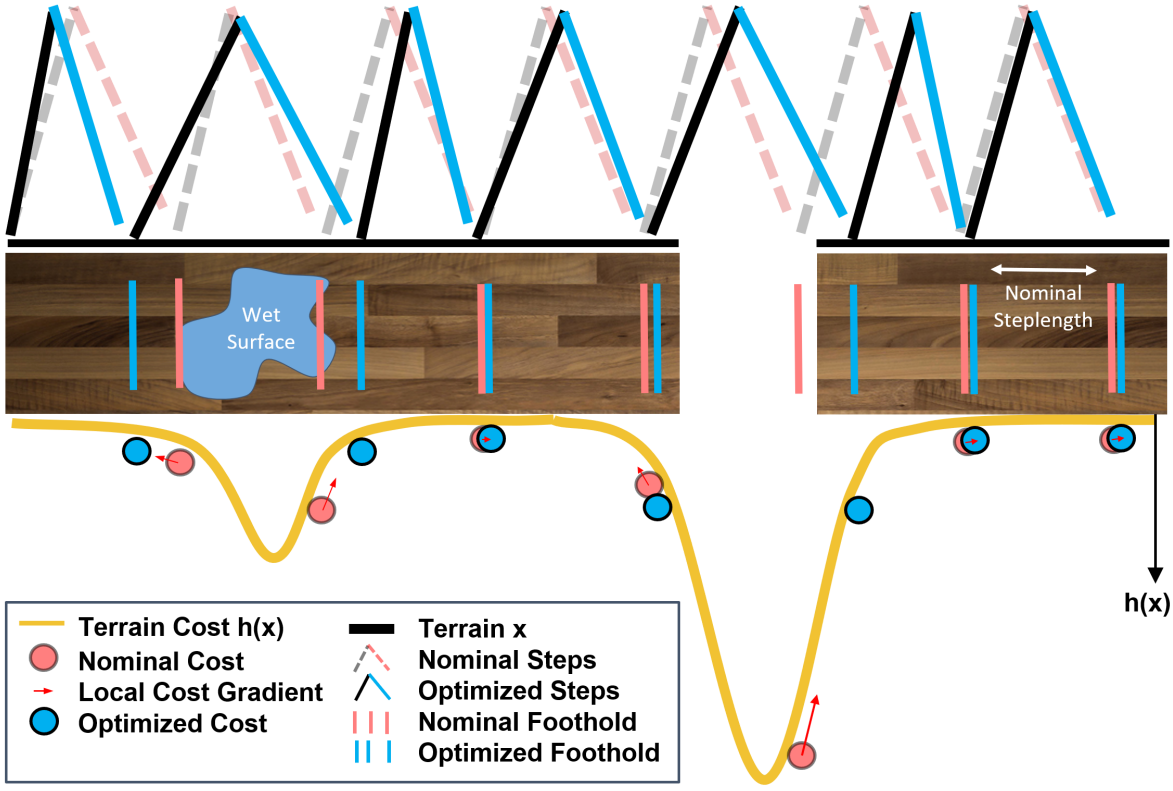

Figure 4.7: Schematic diagram for a legged robot navigating rough terrain, shown with a mirrored terrain cost. Nominal foothold positions are iteratively adjusted toward attainable areas of lower terrain cost by a method of gradient descent. Overall multi-step cost outweighs local cost, evident when Step 4 slightly ascends the slope to assist with the steeper Step 5 descent. The initial model is that of a 2-link compass gait walker over flat terrain.

N-1 steps while fixing the final step.

All three steps of planning and control (footsteps, motion, and feedback) are created concurrently by way of online quadratic programming on the *s.o.s.*, an informed discretization of walking motion by using, and imposing soft penalties on foothold locations directly into our cost function, shown in Fig. 4.7, to push foothold positions toward more suitable terrain. The results shown in Sect. 7.1 support our the claims that our method provides a number of explicit benefits compared to the state-of-the-art: (i) We use a method that does not require solving for time of contact for each incremental control adjustment [41], but instead solves for the gradient of the hybrid step directly. (ii) By discretizing SLQ over the projected dynamics (4.7), our method is able to tune the controllers for both the discrete (4.1) and continuous (4.2) dynamics of the hybrid system

at once. (iii) These projected control updates (4.3) change the nature of the nominal gait, simultaneously adjusting footstep locations and the motion plan, to conform to the terrain map.

While our work has elements similar to existing works, we establish that our method distinguishes itself analytically from similar SLQ approaches in two specific ways:

- We solve for the gradient of *time of contact* t_c with respect to state $\frac{\partial t_c}{\partial \mathbf{x}}$ and control $\frac{\partial t_c}{\partial \mathbf{u}}$ to be used in determining the optimal controller. t_c is represented in this work as the moment in which the foot of a walking system comes in contact with the ground. A fundamental problem with hybrid dynamics is that every small control adjustment comes with a change to t_c, which [41] solve for as an additional optimization parameter. We use a method that does not require solving for t_c for each incremental control adjustment, but instead solves for the gradient of the hybrid step directly. We embed these partial derivatives into our SLQ formulation such that our control solution automatically considers Δt_c for every control adjustment $\Delta \mathbf{u}$. This eliminates the need to solve for additional parameters such as time of contact.

- Rather than discretizing a continuous nominal trajectory into S locally linear segments followed by a discrete event, we discretize the entire hybrid step into one single movement. We do this by embedding a local continuous controller whose parameters $\mathbf{u}$ (gains, etc.) are tuned by the SLQ planner. This approach enables a simple PD controller to govern highly dynamic walking motions. For a finite N-step horizon, this method of linearization transforms a backwards Ricatti control updates from NS to N segments. Our method becomes an N-step foothold optimization against the terrain cost map, combining both foothold selection and motion planning into a unified approach. [94] similarly tuned continuous control parameters (spring constant, etc.), but for LQR control of an infinite time horizon gait transition and not SLQ control for finite time horizon footstep planning as we did. [95] did similar tuning using a deadbeat control approach for footstep placement of a spring-loaded inverted pendulum model, but this system allows "instantaneous

joint angle placement of massless legs" and lacks the complexity of a variable inertia during the continuous spring phase that we consider with the Compass Gait Walker.

4.3 Summary of Unified Framework

In this chapter, we proposed an iterative optimization method that involves real-time planning over a receding horizon to address the problem of dynamic footstep planning for legged systems. Our main technical contribution was a closed-loop implementation of footstep planning and motion control in the context of control effort, facilitated by two supporting techniques. First, the projected hybrid dynamics allow for coupling of the discrete and continuous control actions, considering all motion until impact as a single event and pushes the hybrid control actions toward a common goal. By following the method in Section B.3, we end up with both accurate gradients and very fast computations. The method is well-suited for SLQ and Riccati-like optimization techniques. Second, the terrain was treated as a cost map based on desirability of foothold positions. By using soft constraints, we allow the hybrid system to choose its terrain while balancing control effort through an optimization scheme utilizing gradient descent.

Chapter 5

Energy Management Through Footstep Selection

While the unified framework in Chap. 4 can adjust foothold positions, so far we have no demonstrated way to generate the initial stable nominal trajectory, the basis for all iterative planning adjustments. As discussed in Chap. 2, many works initialize their system with a known map of the terrain and a kinematically-feasible seed. We consider the terrain map must be generated online, and therefore all planning must be done online, to include determining an initial path over rough terrain. Creating a dynamically feasible plan over rough terrain is not trivial and this chapter discusses how a passive system can select foothold positions by managing its system energy.

We introduce a method that uses foot contacts to steer the body COM along energy profiles over rough terrain. We use a simplified model to illustrate how leg placement affects system energy and movement. This work derives two boundaries that are fundamental to walking: the first separates walking from falling while the second subdivides walking into slowing down and speeding up.

It is the authors' understanding that the following contributions are new to dynamic walking over rough terrain. We demonstrate how energy-centric foot placement can permit a passive system to move over rough terrain, rather than rejecting terrain through feedback. Through an analysis of the post-impact energy, we show that the reachable

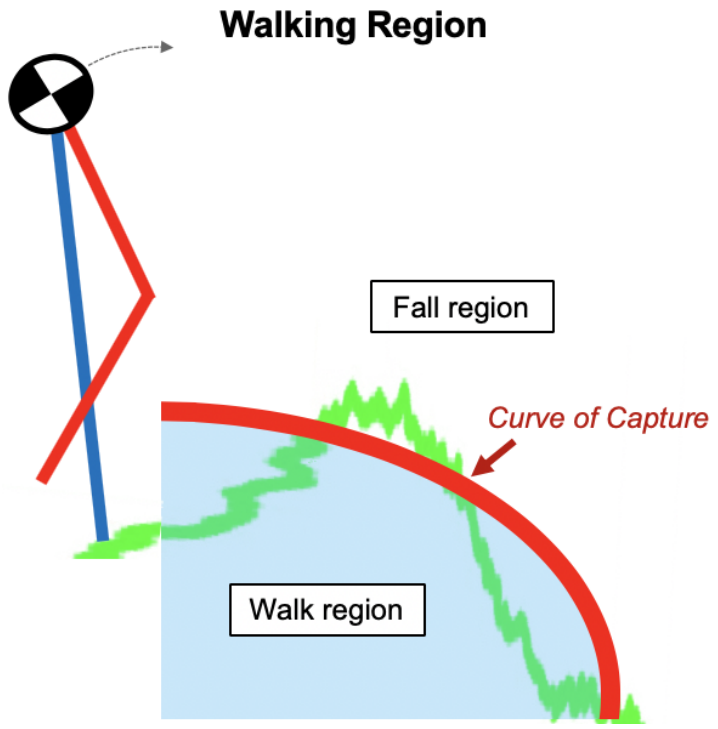

Figure 5.1: Walking Region for a Passive System. The space of kinematically-feasible footstep regions can be divided by the *curve of capture* into regions that either promote dynamic walking or falling backwards. A simple nonlinear inverted pendulum is used to express this method.

foothold positions can be divided into regions that permit walking or cause falling, shown in Fig. 5.1. Step placement is not simply determining whether the system will fall or not, but also where it can step to ensure future steps have enough energy to succeed. It will then be shown that the walking region can be further subdivided into sectors that either assist, maintain, or hinder walking. It will be shown that knowledge of these sectors are imperative to the accumulation or dissipation of energy, even if the terrain map is not perfect.

5.1 *Curve of Capture*

Pratt [52] introduced the idea of capture point (CP) to the LIP model. The core use was foot placement as a means of push recovery. Other works extended the idea of CP to the NIP [76], [96] as a target for foot placement to cease body motion. Instead of just a theoretical means of stopping motion, the CP can be defined as a boundary between walking and falling backwards as shown in Fig. 5.2. This work uses the idea of CP as a

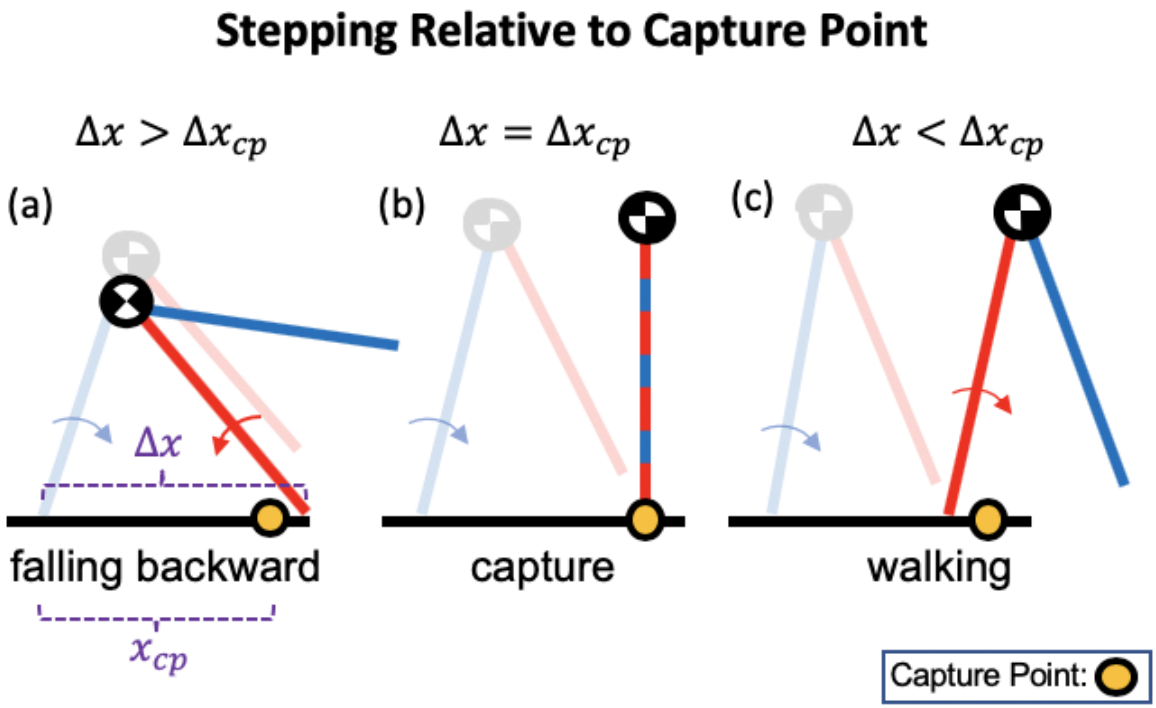

Figure 5.2: Stepping Relative to Capture Point x_{cp}. (a) Overstepping the CP results in falling backwards. (b) Stepping directly onto a CP results in the NIP stopping at the unstable equilibrium point. (c) Stepping before the CP results in a walking behavior.

boundary for successful walking over uneven terrain.

Computing the NIP CP is relatively straight forward, but there is no analytical solution. It begins with an understanding that capture occurs when the post-collision energy is equal to the max potential energy of the system ($E^+ = \mathcal{U}_{max}$). This is possible because of the dissipation of energy through the impact map M while accounting for a change of foot height Δz. The local change between foot origin translation position and height in the global frame x, z is described by (3.19) as

$$\begin{aligned} \Delta x &= x^+ - x^- = -l\left(\sin\left(\theta_1^- + \theta_2^-\right) + \sin\theta_1^-\right) \\ \Delta z &= z_o^+ - z_o^- = l\left(\cos\theta_1^- + \cos\left(\theta_1^- + \theta_2^-\right)\right) \end{aligned} \tag{5.1}$$

Assuming we know the energy of a step E^-, we need to determine foot placement to dissipate all of the kinetic energy at the unstable equilibrium point $\theta^+ = 0$. The Kinetic

and Potential energy, $\mathcal{T}$ and $\mathcal{U}$, respectively, can be written as

$$\mathcal{T} = \frac{ml^2}{2}\dot{\theta}_1^2 \tag{5.2}$$

$$\mathcal{U} = mgl\cos\theta_1 \tag{5.3}$$

Using (5.2),(5.3) and (3.21), we can determine the post-collision energy relationship in terms of pre-collision components

$$\begin{aligned}\Delta E &= \left(\mathcal{T}^+ + \mathcal{U}^+\right) - \left(\mathcal{T}^- + \mathcal{U}^-\right) \\ mgl - E^- &= -\frac{ml^2}{2}\left(\dot{\theta}^-\right)^2 \sin^2\theta^- - mgl\left(\cos\left(\theta_1^- + \theta_2^-\right) + \cos\theta_1^-\right).\end{aligned} \tag{5.4}$$

Since the system is passive, we can compute the impact velocity as a function of pre-collision system energy E^-

$$\dot{\theta}_1^- = -\sqrt{\frac{2}{ml^2}\left(E^- + mgl\cos\theta_1^-\right)}. \tag{5.5}$$

Substituting (5.5) into (5.4) yields an expression that is dependent only upon pre-collision configuration

$$mgl - E^- = -\left(E^- - mgl\cos\theta_1^-\right)\sin^2\theta_1^- - mgl\left(\cos\left(\theta_1^- + \theta_2^-\right) + \cos\theta_1^-\right). \tag{5.6}$$

By substituting (5.1) into (5.6)

$$mgl\left(1 + mgl\Delta z\right) - E^- = \left(mgl\cos\theta_1^- - E^-\right)\sin^2\theta^-, \tag{5.7}$$

it becomes clear that there is a matching body position for a given toe height Δz, assuming only walking configurations ($-\frac{\pi}{2} < \theta^- < \frac{\pi}{2}$ and $E^- > E^+$). Since (5.7) does not have a closed-form solution, this work uses a least-square polynomial regression of sampled points in the space to determine the relationship between energy and configuration that

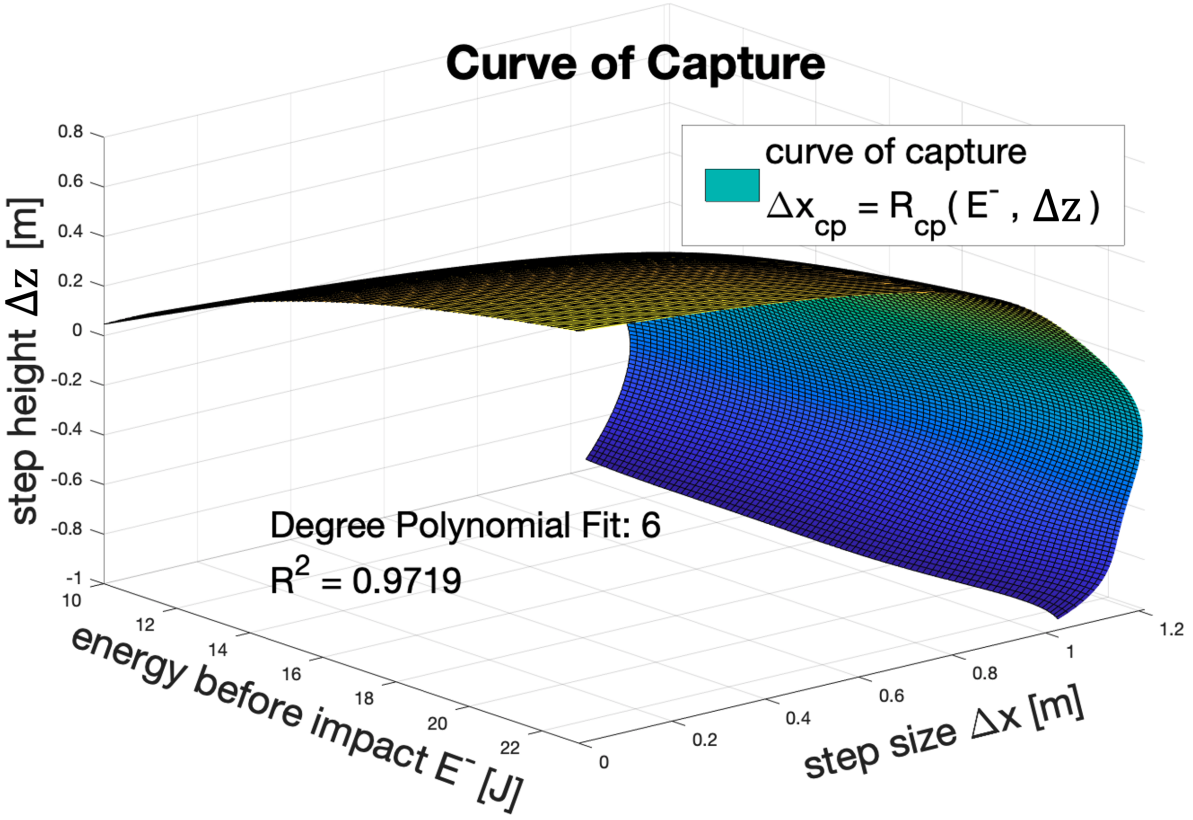

Figure 5.3: *Curve of Capture*

reaches the CP, whose coefficients are shown in Appendix C.

$$\Delta x_{cp} = R_{cp}\left(E^{-}, \Delta z\right). \tag{5.8}$$

The shape of the regression is shown in Fig. 5.3. As energy increases, the biped can step higher and further prior to reaching the CP.

This curve provides a very intuitive meaning: everything to the left of the curve results in a walking behavior and everything to the right results in falling backwards as seen in Fig. 5.4. As a robot is walking and looking for a safe place to step that ensures forward motion, this curve creates the boundary for dynamically-safe walking behavior.

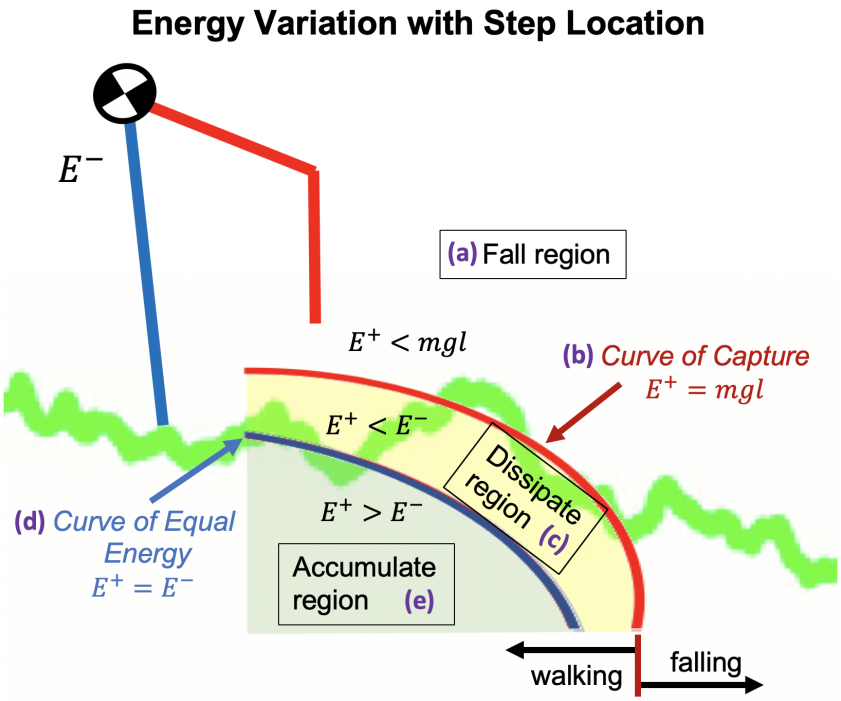

Figure 5.4: Energy Variation with Step Location. (a) Stepping on terrain in the void beyond the *curve of capture* means the system does not have the energy to reach the vertical position and results in falling backwards. (b) Stepping onto the *curve of capture* is the exact energy required to stop at the unstable equilibrium point. Stepping anywhere below this curve results in walking. (c) Stepping between the dissipate region results in walking, but a loss in energy. (d) Stepping directly onto the *curve of equal energy* means the kinetic energy lost through impact is equal to the potential energy gained. (e) Stepping below, into the accumulate region results in speeding up, where subsequent steps have more local energy than the previous step.

5.2 *Curve of Equal Energy*

Another important step location for a passive system is the location where local energy is conserved. Consider a biped walking down a slope. The system gains potential energy by walking down the slope, but loses kinetic energy at impact. Short steps result in a quick-stepping behavior that increases energy ($E^+ > E^-$). Stepping too far, but not yet to the CP, results in the system losing energy ($E^+ < E^-$), but still takes a step. There is a specific step length that corresponds to a steady-state gait ($v_{n+1} = v_n$) or ($E^+ = E^-$) (equivalent to a fixed point on a return map). By setting $\Delta E = 0$, we arrive at the expression

$$mgl\Delta z = \left(mgl\cos\theta^- - E^-\right)\sin^2\theta^-. \tag{5.9}$$

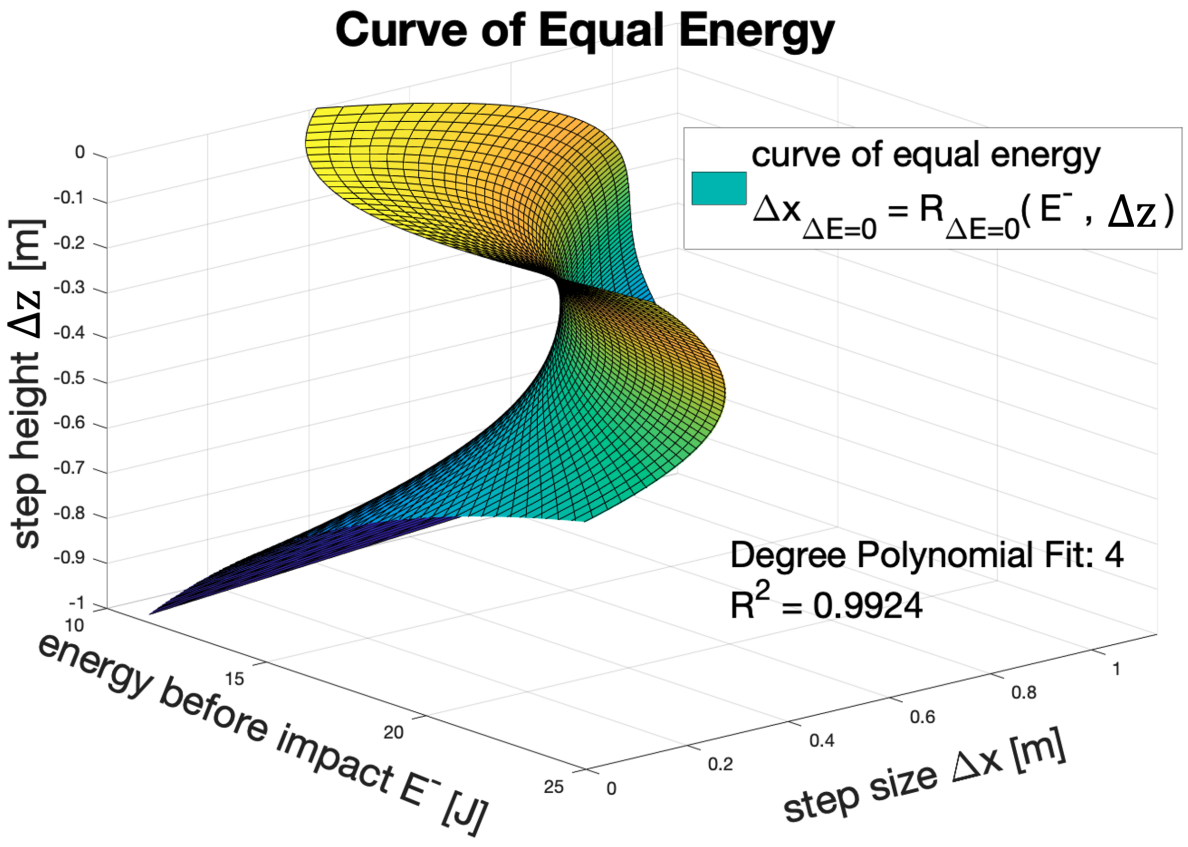

Figure 5.5: *Curve of Equal Energy*

Similar to (5.7), this work uses a least-square polynomial regression to determine the relationship between energy and configuration that reaches the equal energy, whose coefficients are shown in Appendix C.

$$\Delta x_{\Delta E=0} = R_{\Delta E=0}\left(E^{-}, \Delta z\right). \tag{5.10}$$

The shape of the regression is shown in Fig. 5.5.

This curve denotes an important concept: step to the left to speed up or to the right to slow down. If the robot must speed up to overcome an approaching obstacle, it can accumulate energy by taking several steps to the left of the curve. Likewise, if moving too fast, step to the right of the curve to dissipate energy.

The regions separated by the *curve of capture* and *curve of equal energy* can be depicted by the phase portrait [97] of a simple pendulum (see Fig. 5.6). Stepping directly onto the *curve of equal energy* (d) results in maintaining the same energy level in the

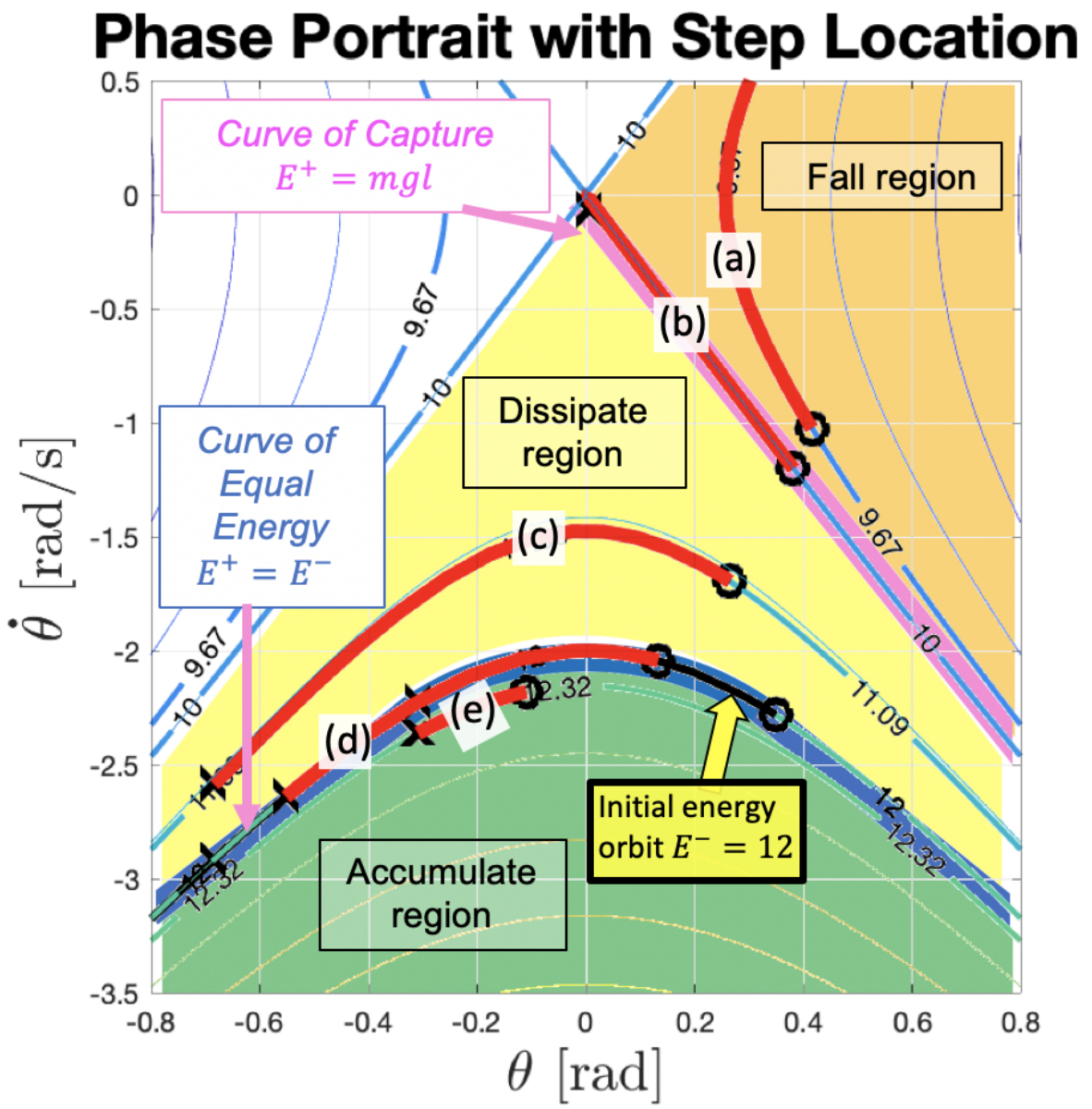

Figure 5.6: Phase Portrait with Step Location. These red paths represent what happens when stepping into the regions shown in Fig. 5.4 with $mgl = 10$. Contours represent energy levels. (a) Overstepping the *curve of capture* results in oscillation (falling backwards). (b) Stepping onto the *curve of capture* takes the system up the homoclinic orbit to the separatrix. (c) Losing energy with a long step, but still walking. (d) Stepping onto *curve of equal energy* returns to same rotation orbit. Post-collision energy matches pre-collision energy. (e) Gaining energy with a short step.

traveling orbit. Stepping short/long result in energy increase/decrease ((e)/(c)). Stepping on the *curve of capture* (b) puts the system directly onto the homoclinic orbit which takes it to the unstable equilibrium point (separatrix). Stepping past the *curve of capture* (a) results in an pendulum-like behavior (a walker falling backwards).

5.3 Summary of Energy Method

In this chapter, we separated the kinematically-feasible step region into areas of dynamic relavance using a *curve of capture* and *curve of equal energy*, defined in Sections 5.1 and 5.2. In Chap. 7, we will present results for these methods using a passive nonlinear inverted pendulum (NIP) model walking over very rough terrain. Next, in Chap. 6, we will demonstrate how this method can be used as a seed for the unified framework.

Chapter 6

Seeding Unified Planning through Energy Management

This work combines the previous two seemingly-disparate ideas from Chap. 4 and Chap. 5 into one single unified framework. For an underactuated system with mass in the swing leg, we identify the region in which footsteps can be safely placed to ensure walking continues over rough terrain using a controlled version of the *curve of capture*. As the robot moves forward and new terrain is perceived, we use this placement strategy to identify new foothold positions that satisfy both physical and dynamic constraints. This strategy is stitched into the fabric of the unified framework in the form of an initial guess for the iterative methods to optimize as shown in Fig. 6.1. Specifically, this work fills in the missing element of the unified framework, allowing the system to dynamically adapt a system's behavior when unexpected events occur.

6.1 Energy Technique applied to system with massive legs

In Chap. 5, we generated a method for guaranteeing safe walking by selecting footholds behind the *curve of capture*. This technique was so far constructed only for a passive NIP walking model, yet it does not yet account for a system whose swing leg has mass and

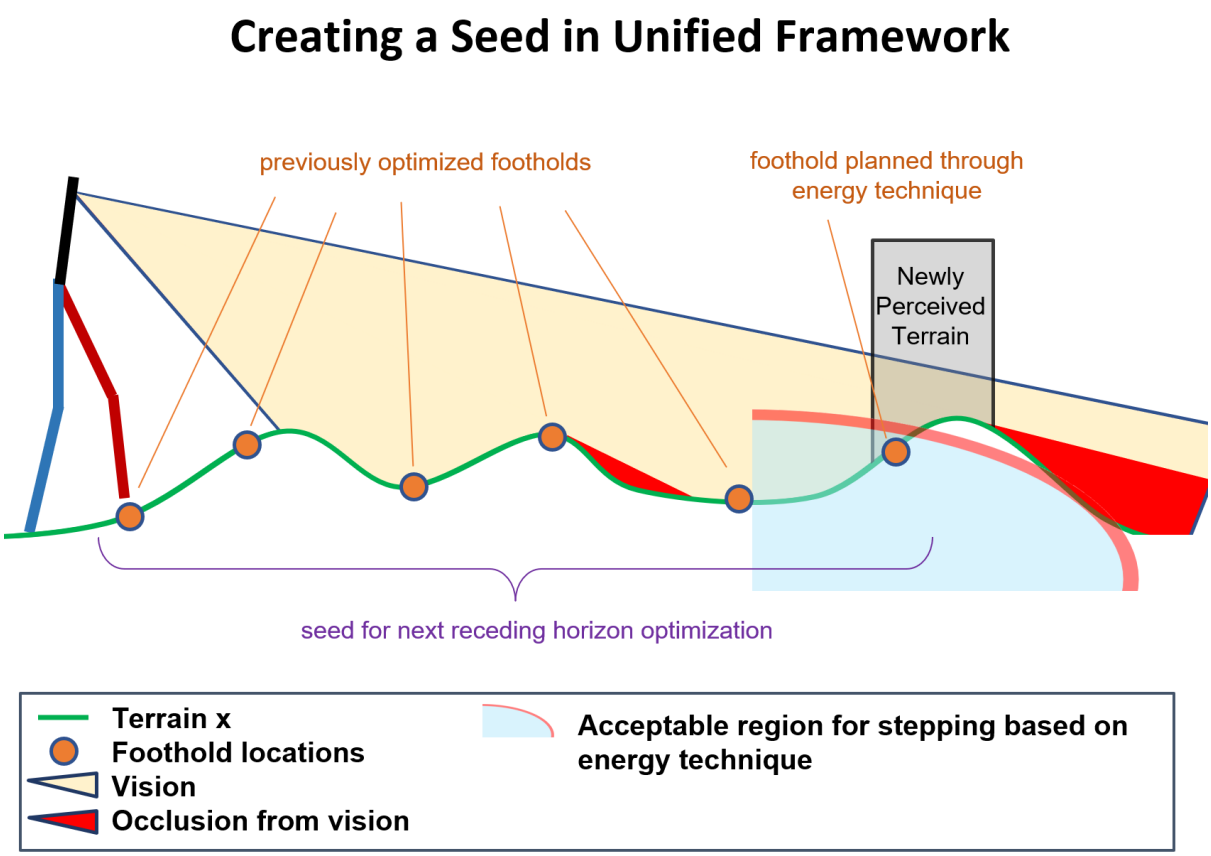

Figure 6.1: Creating a Seed for Newly Perceived Terrain. The first 5 steps are the results of the previous unified optimization. The sensor detects new terrain and an initial plan for the newly perceived terrain is created. The two plans are stitched together to form the seed for the next receding horizon optimization. This process occurs at least once per physical robot step.

inertia such as the CGW and 5-link models. If the swing mass and inertia are minimal relative to the body such as in the robot Cassie [98], the LIP and NIP ROMs have proven to be acceptable [99] since legs can be placed very quickly. But for most bipedal systems, the swing leg movement has great effect on the COM trajectory and timing for foot placement. Therefore, we have to take into account the mass and inertia of the swing leg, as well as the control strategy being used to swing the leg.

We will model the trajectory of the system COM as it is affected by the control strategy. First, we establish the coordinate system for the system COM, which can be determined by its kinematics at any given moment as shown in Fig. 6.2. Our COM ROM has two key differences than that of the NIP: i) The legs are no longer constant when

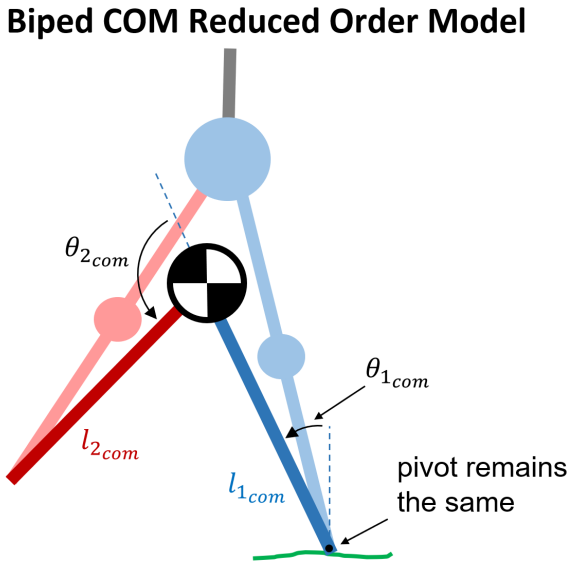

Figure 6.2: Biped Reduced Order Model. The system becomes the same as the NIP model but with variable length legs.

the internal joints are moving. As the configuration changes, the COM position moves relative to the stance foot pivot. ii) The system has a variable moment of inertia about its COM. Depending upon how the legs are positioned, the moment of inertia about the COM changes. When the internal joints stop moving legs stop moving, the system becomes a rigid body with a constant moment of inertia rotating about the stance foot.

A hybrid step occurs in 3 phases with an additional phase to achieve the CP, as shown in Fig. 6.3:

a) Swing leg moves according to policy **U** toward desired step length x^d_{step}. COM leg lengths vary according to kinematics.

b) System becomes rigid and rotates about the stance foot as a NIP model.

c) System impacts the ground with inelastic impact and swing foot becomes new pivot.

d) Swing leg moves according to policy **U** toward some nominal step length. If it has too little energy, the system falls backward. If it has just enough energy, it achieves capture as a rigid system. If it has too much energy, the system resumes walking.

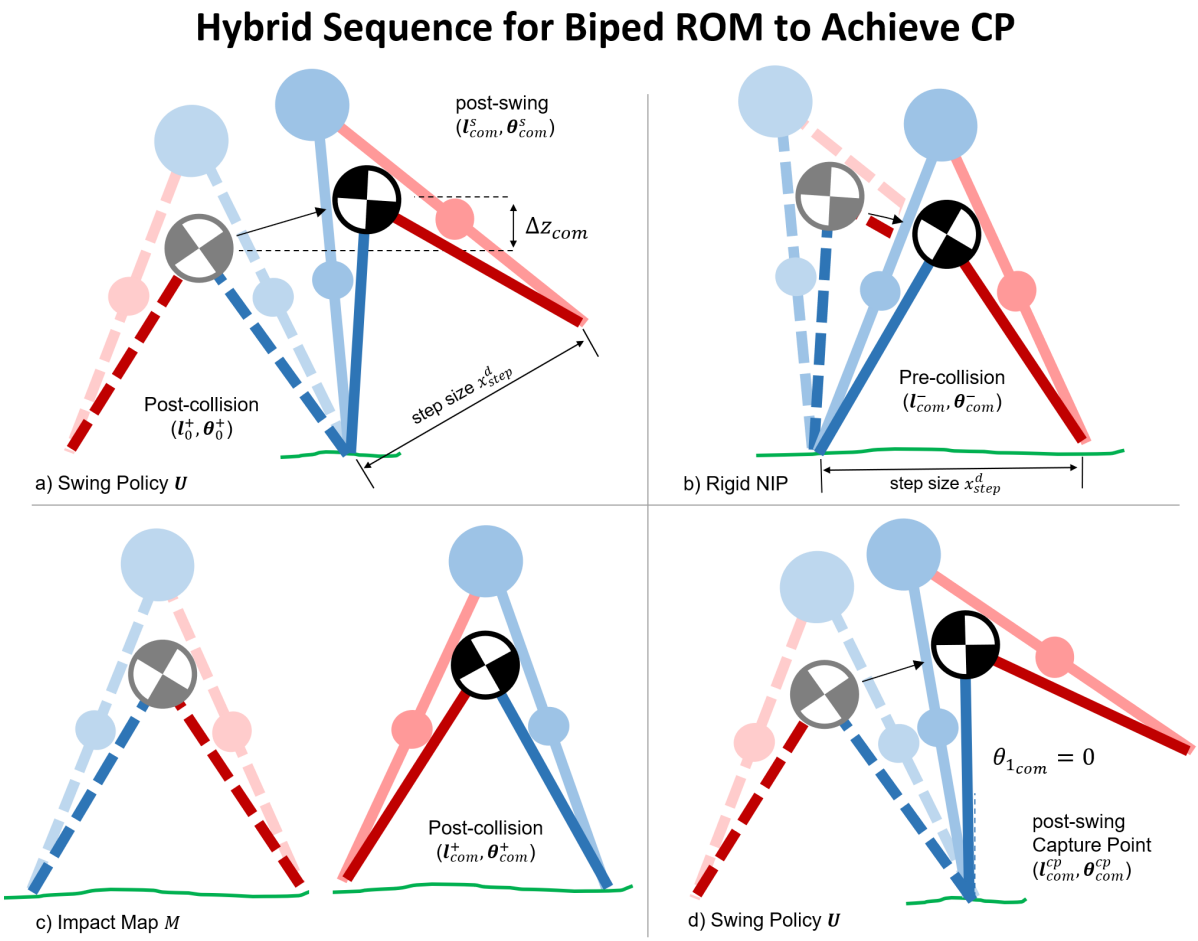

Figure 6.3: Hybrid Sequence for Biped Reduced Order Model to Achieve Capture Point. a) Massive swing leg moves according to policy **U** b) The system moves as a rigid system, like a NIP. c) Impact MAP M. d) Massive swing leg moves according to policy **U** to achieve capture

6.1.1 Energy Method applied to CGW model

Consider the CGW (see Fig. 3.7) with a strategy for PD control about its inter-leg angle as well as gravity compensation and Coriolis canceling about its hip

$$\ddot{\theta}_2^d = -K_{p_2}\left(\theta_2 - \theta_2^d\right) - K_{d_2}\dot{\theta}_2. \tag{6.1}$$

Derived from (3.4), (3.22), (3.23), and (3.24), we have

$$\tau_2 = ml_{thigh}^2\ddot{\theta}_2^d + m(l_{calf} + l_{thigh})l_{thigh}\sin(\theta_1 - \theta_2)\dot{\theta}_1\dot{\theta}_2 + ml_{calf}g\sin(\theta_2). \tag{6.2}$$

Together, (6.1) and (6.2) make up an example policy **U** discussed in phases a-d above.

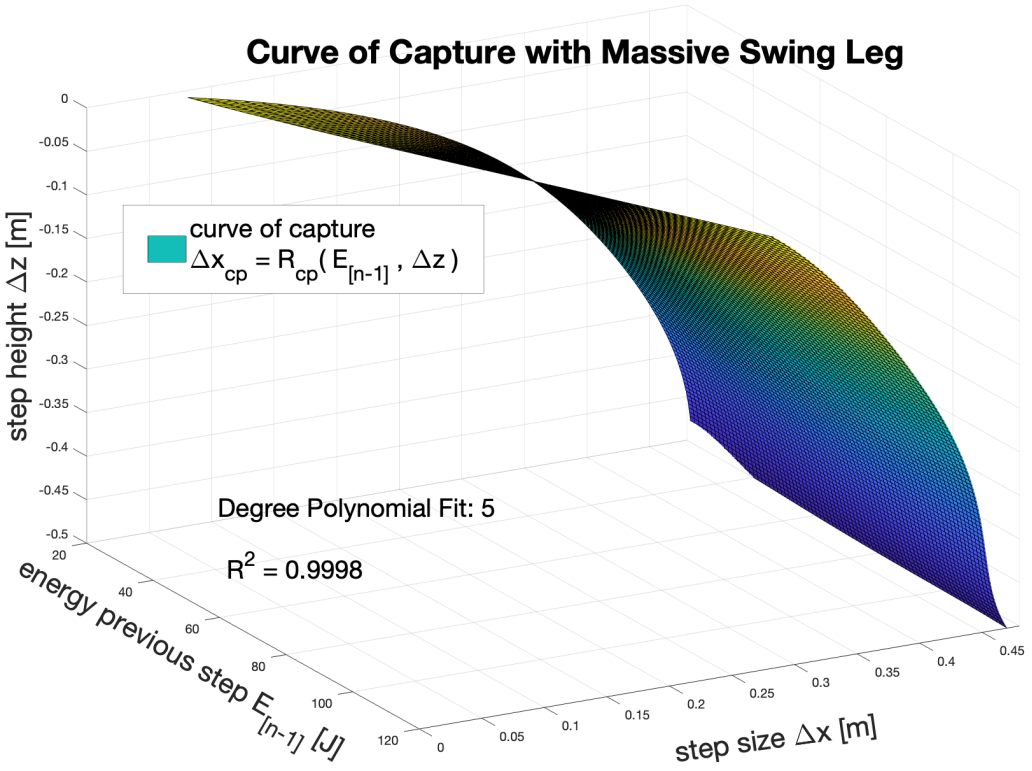

Figure 6.4: *Curve of Capture* for CGW given policy **U**.

Similar to (5.7) and (5.9), this work uses a least-square polynomial regression to determine the relationship between energy and configuration that reaches the capture point, whose coefficients are shown in Appendix C.

$$\Delta x_{\Delta cp} = R_{\Delta cp}\left(E_{[n-1]}, \Delta z\right). \tag{6.3}$$

The shape of the mesh from the regression is shown in Fig. 6.4.

As we built this model, it became very apparent the significance of mass distribution within the system. Since the system has no massive torso to bend, all energy must be then generated from the swing leg. If the swing leg has a high COM ($l_{c_2} << a_2$), then the leg is unlikely to produce sufficient torque to swing the heavier hip mass over the CP. On the other hand, if the COM is low on the leg ($l_{c_2} \approx a_2$), then with sufficient hip torque the leg can more easily propel the system forward. Therefore, by moving the COM lower on the legs, we can expand the *curve of capture*.

6.1.2 Energy Method applied to 5-link walker model

For more complex systems such as the 5-link walker, similar regressions can be built, but they require additional inputs to designate the length of the legs before and after collision as indicated in Fig. 6.5. In this figure, the swing leg impacts with a length less than $l_{calf} + l_{thigh}$. It can be seen that by bending the knee forward, the CG moves forward and when the swing leg becomes the new pivot, the after-impact moment arm is shortened. As the CP moves to the right, the biped can therefore walk using step areas that would have previously been impossible with a straight leg at impact.

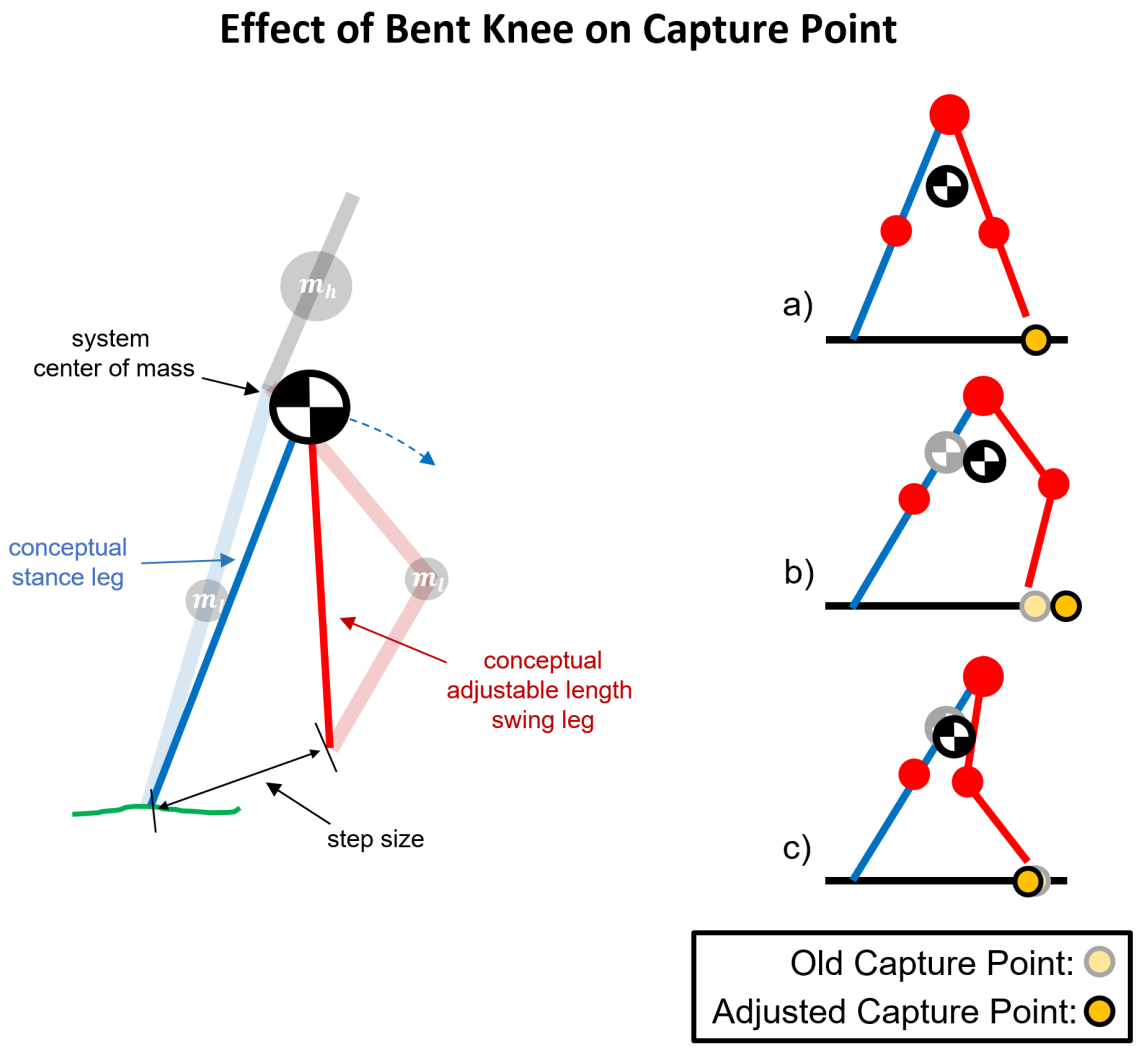

Figure 6.5: Effect of Bent Knee on Capture Point. Once the system joints stops swinging, the system can be reduced to a system of a point mass and two legs of varying length. In this example, we look at what happens to the CP by varying the length of the swing leg before touchdown. a) Swing leg straight. Baseline for comparison. b) Bending the swing leg forward while retaining the same step length pushes the CP to the right. c) Bending the swing leg backwards while retaining the same step length has only a mild affect on the CP.

6.2 Unified Planning Architecture - Revisited

In this section, we show how the energy management can be used to seed the unified planning construct from Chap. 4. To illustrate the seeding process, we will now describe the seeding process shwon in Fig. 6.6. For known terrain, we begin with a plan (1). For terrain that is newly perceived (no step plan yet exists), we create footstep plans (2) using the energy technique from Chap. 5. (1) and (2) are combined into a sequence of N steps,

in which each step is linearized along the projected hybrid dynamics from Chap. 4 to form the discrete seed (3) necessary to initialize the trajectory optimization. These N steps become the N discrete segments of the SLQ optimization scheme. Through a series of forward and backward passes, the scheme converges upon a locally-optimal solution which becomes the N step plan (4). The first step of the plan (6) is passed to the controller for execution. The remaining $N-1$ steps of the plan (5) are returned to the seeding process, now known as "previously-optimized steps."

This seeding process is demonstrated on hardware, with results shown in Chap. 7.

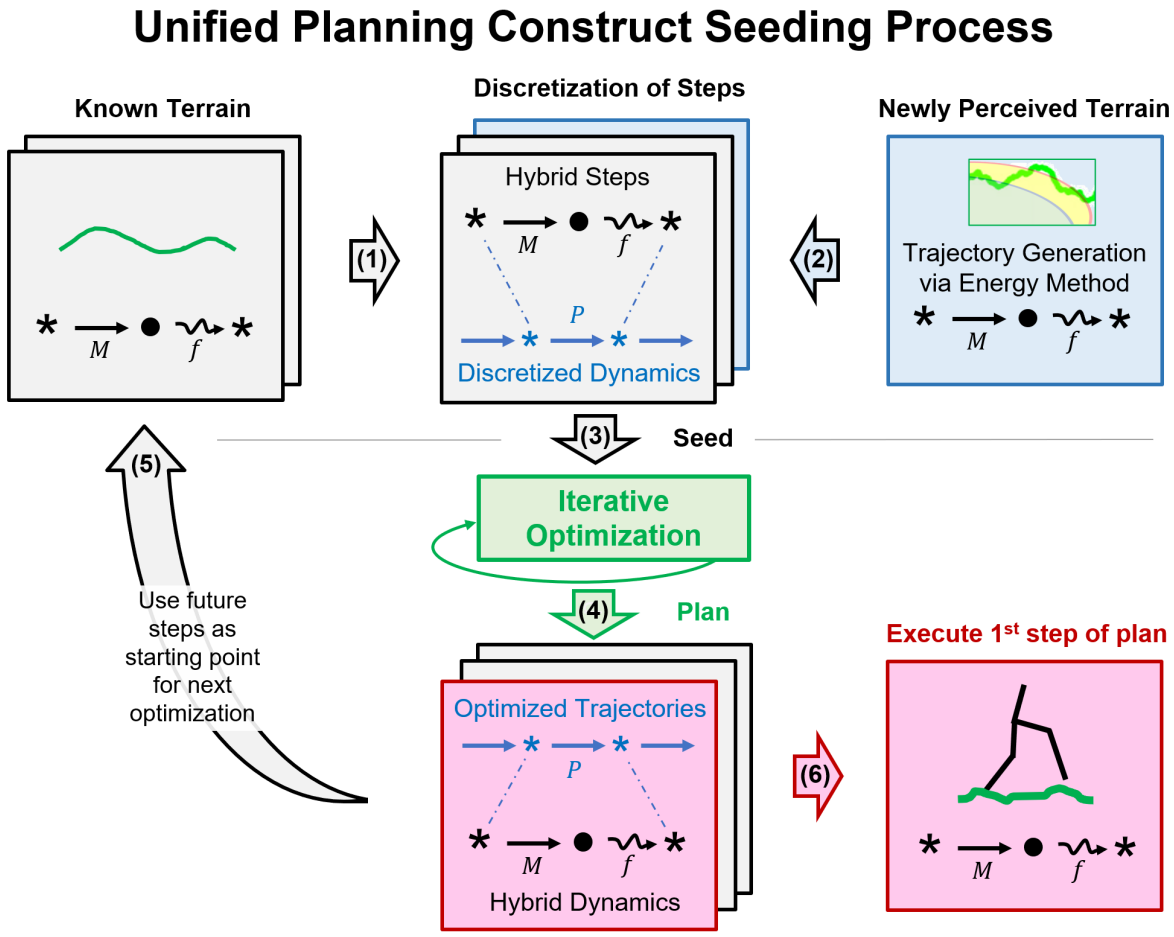

Figure 6.6: Unified Planning Construct Seeding Process. Hybrid trajectories over known terrain (1) are joined with on-line trajectory planning for newly perceived terrain (2). The trajectory is then discretized into N hybrid steps to form the "seed" (3) for a once-per-step optimization scheme. This iterative optimization adjusts foothold placements to conform to a terrain cost map, while simultaneously updating the low level feedback controllers and their corresponding motion plans for all steps on the planning horizon. The optimized plan (4) is separated into $N-1$ steps to become a starting point (5) for the next seed and the piece to presently execute. The walking system executes the first step of the plan (6), while the remaining steps are sent back to become part of the seed for the next round of planning.

Chapter 7

Results

7.1 Unified Results over Flat Terrain

In this section, we complete an example of a Compass Gait Walker (CGW) walking over a terrain map. While simplified walking models [72], [95] could greatly reduce the computation time while fairly approximating the system dynamics, the assumption of massless legs oversimplifies most humanoid robots. Thus, our system has masses at the hip and knees with continuous PD-controlled torque in the hip and discrete toe-off impulses. We feel this combination properly demonstrates the use of SLQ feedforward control and feedback gains to optimally mix the discrete and continuous control efforts in a way to bring a controlled hybrid system to a locally-optimal and dynamically-feasible path.

The system begins passively walking down a slope, but adds control effort to minimize the objective function (4.20) while maintaining dynamic stability. Given a nominal gait pattern, the nonlinear hybrid controller iteratively modifies individual step sizes to secure foothold positions in areas of lower terrain cost for a 10-step sequence. The gradient of the local cost polynomial $p_{[n]}$ pushes the foothold location away from undesirable areas as shown in Fig. 7.2. The step sizes adjust as the individual costs are driven toward a local minima as shown in Fig. 7.3. The dynamics are preserved as the system modifies its stride length.

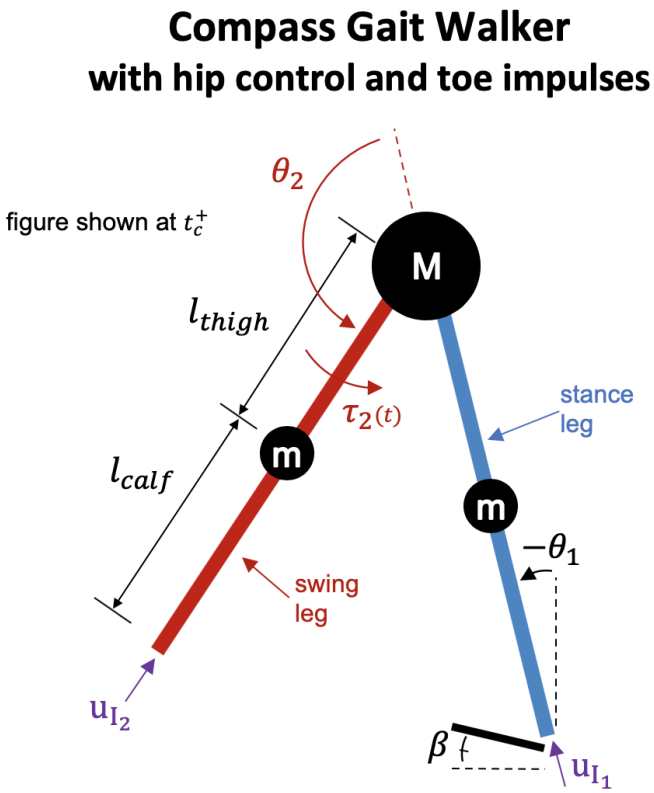

Figure 7.1: Compass Gait Walker with hip torque and toe-off impulses. Parameters: $m = 5kg$, $M = 10kg$, $l_{thigh} = l_{calf} = 0.5m$. Slope of terrain at stance foot: $\beta = 3°$. Front toe impulse u_{I_1} occurs at t_c^- time instantaneously before touchdown (remove energy from system) and rear toe impulse u_{I_2} occurs at t_c^+ time instantaneously after touchdown (add energy to system). Together, these impulses form the control actions to the discrete collision map M (4.1). Hip torque $\tau(t) = \begin{bmatrix} k_{p_1} & k_{p_2} & k_{d_1} & k_{d_2} \end{bmatrix} \begin{bmatrix} \bar{\theta}_1 & \bar{\theta}_2 & \dot{\bar{\theta}}_1 & \dot{\bar{\theta}}_2 \end{bmatrix}^T$ offers continuous control throughout the step. Together, these form the admissible once-per-step control actions $\mathbf{u}_n = \begin{bmatrix} u_{I_1} & u_{I_2} & k_{p_1} & k_{p_2} & k_{d_1} & k_{d_2} \end{bmatrix}^T$ shown in (4.3) and used in (4.6).

For gradient computation, the analytical integration of (B.9)-(B.11) to generate the state and control gradients of the projected hybrid dynamics is on the computational complexity order of that of Forward Difference methods, but with better accuracy than Central Difference schemes when spacing h_i is well-tuned for each state $\mathbf{x}_i$. Additionally, with the use of symbolic computations of the terms in (B.9)-(B.11), common terms are not computed more than once, lending to faster overall speed. Our analytical method is 7.4× and 14.0× faster than the Forward and Central Difference algorithms, respectively, to compute the hybrid gradients. See Table 7.1 for the comparison of a CGW taking one step. The times referenced were calculated in MATLAB over an average of 10,000 runs on a computer with a 3.1 GHz Intel Core i7 processor and 16GB 2133 MHz ram.

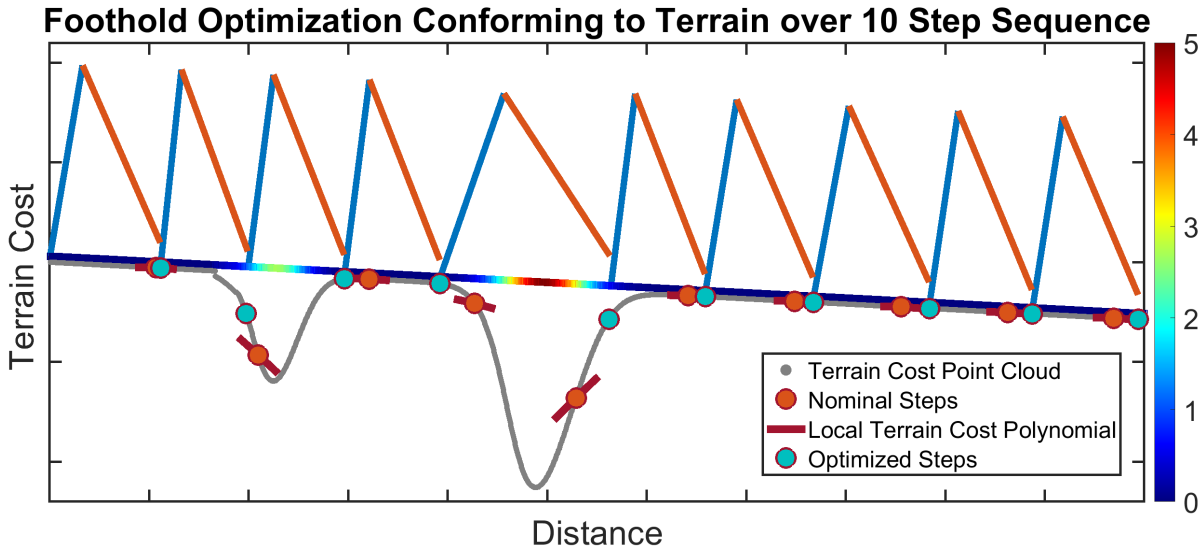

Figure 7.2: Foothold optimization conforming to terrain cost map using gradient descent of local polynomials. Optimized results are shown after 4 iterations of SLQ. The controller modifies the step size to dynamically conform to the terrain, regaining the passive gait by Step 10.

Table 7.1: Computational Complexity, Accuracy, and Time for Numerical Differentiation Schemes Compared with the Analytical Integration Method Shown in B.3

∇ Method	A^*	B^*	Accuracy	Comp Time
Central Diff	$2n(n-1)$	$2nm$	$O(h^2)$	12.99 ms
Forward Diff	n^2	$n(m+1)$	$O(h)$	6.85 ms
Analytically	$(n-1)^2$	$(n-1)m$	$< O(h^2)$	0.93 ms

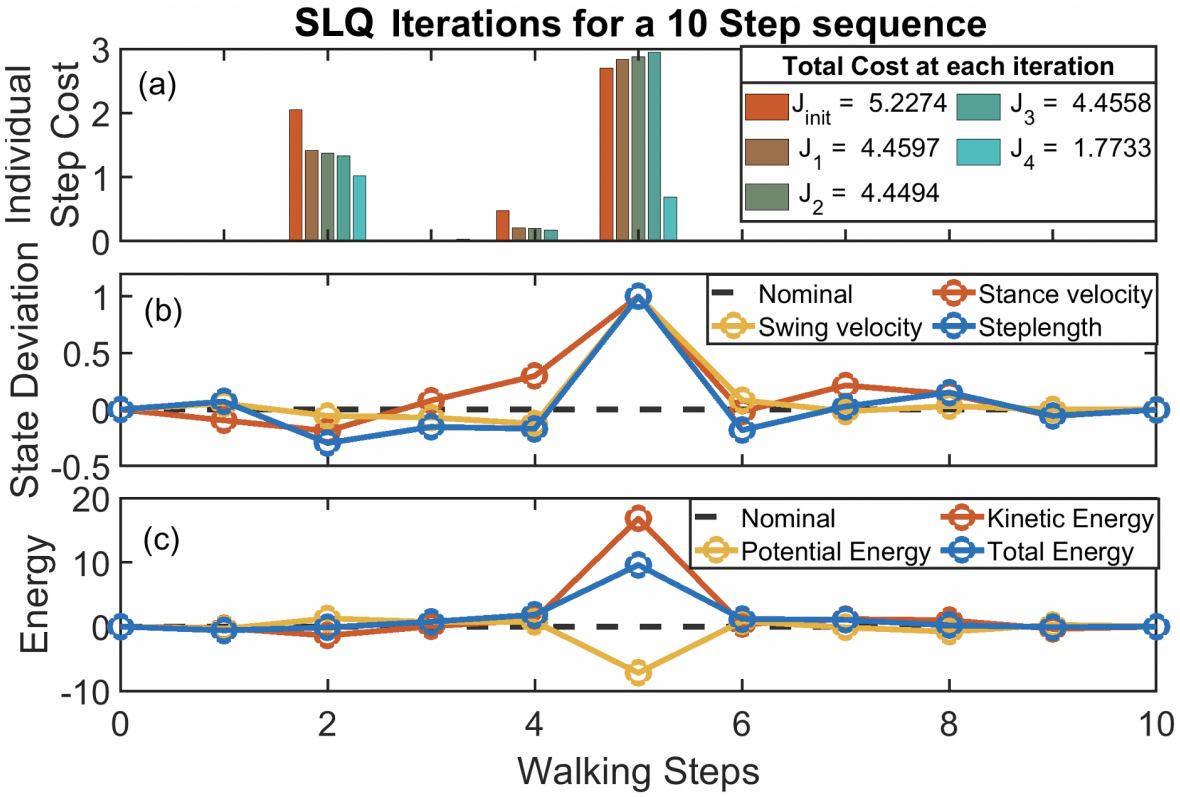

Figure 7.3: SLQ Iterations for a 10 Step Sequence. (a) Footstep cost converging in 4 iterations of SLQ. In area of highest terrain cost (matching terrain in Fig. 7.2), local step cost increases slightly before dropping off drastically in iteration 4. (b) State deviations from nominal scaled by max deviation. This figure shows several shorter steps before a long step over a region of high local cost. Full state recovery happens at step 10. (c) Energy deviation from nominal. We see an increase in the KE of the system as it plans to take a bigger step.

7.2 Energy Management Results over Rough Terrain

This test consists of accumulating energy to cross a gap in the terrain (see Fig. 7.4). Initially, the reference gap *curve of capture* is well below the lip of the far side of the gap. For 7 steps, the robot places its foot to the left of the *curve of equal energy*, adding local energy to the system. At each step, the gap *curve of capture* expands, pushing upward. By step 9, the gap *curve of capture* pushed over the far side of the gap, ensuring the system could safely step. Depicted by the transition from $\mathbf{q}_9^-$ to $\mathbf{q}_{10}^+$ in the corresponding phase portrait, the system lost $4.5J$ of energy crossing the gap, indicating that the capture point energy was approximately $14.5J$. The system did not have enough energy to cross the gap in the first 8 steps. Here is a video link of these results.

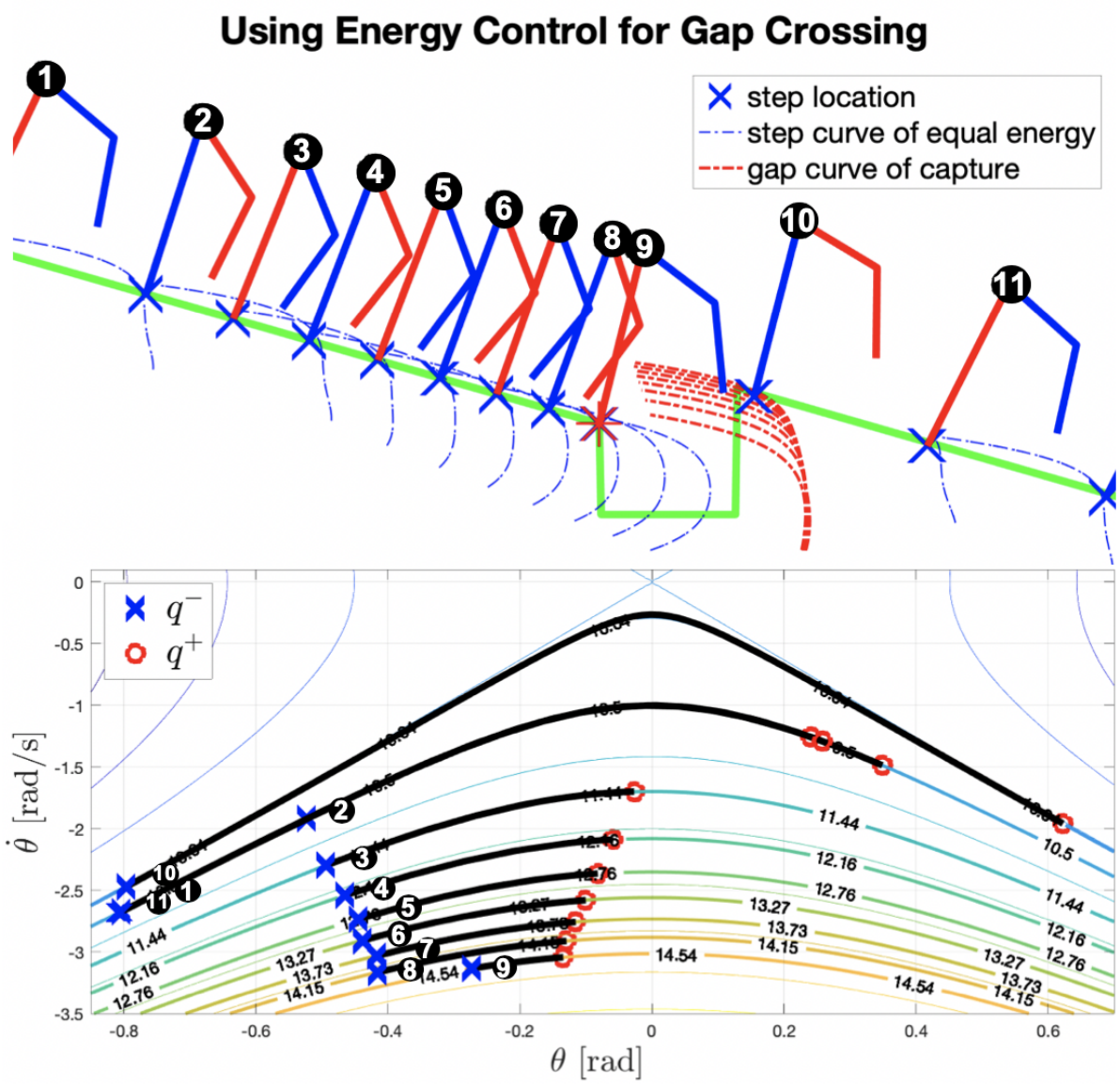

Figure 7.4: Using Energy Control for Gap Crossing. $E_{min} = mgl = 10J$. For step (1), the robot starts with energy $= 10.5J$ at steady state. During steps (2)-(8), the robot steps to the left of the *curve of equality* to accumulate energy. As the robot accumulates energy, the *curve of capture* across the gap incrementally pushes up. At step (9), the biped has just enough energy to cross the gap ($E^+ = 10.05J$). For steps (10)-(11), the biped returns to step locations that result in the nominal walking energy.

7.3 Unified Planning Hardware Validation

7.3.1 The Robot

In order to validate this optimization scheme, We built a 5-link legged robot, shown in Fig. 3.8. We use Hebi actuators for internal joint activation, with *X8-3* modules at the knees and *X5-4* modules at the hips. The heavier *X8-3* modules give us the desired

characteristic of a massive swing leg. The system is mounted to a boom in order to restrict motion to the plane, yet the robot is free to rotate about the joint connecting the robot. In order to allow the robot to move more freely under its own weight, the mass of the boom is offset by a counterbalance. We feel this system is true to the 5-link biped model shown in Fig. 7.5.

The DH parameters and mass properties for our biped are defined in Table 7.2. All units are standard SI. The corresponding link teardown is shown in Fig. 7.6.

Table 7.2: Denavit-Hartenberg Parameters and Mass Properties for 5-Link Robot

Link	a_i	α_i	d_i	θ_i	m_i	l_{c_i}	j_i	Description
1	0.299	0	0	θ_1	0.256	0.157	0.10	stance calf
2	0.279	0	0	θ_2	0.822	0.064	0.22	stance thigh
3	0	0	0	θ_3	2.865	0.085	0.40	torso
4	0.279	0	0	θ_4	0.822	0.215	0.22	swing thigh
5	0.299	0	0	θ_5	0.256	0.142	0.10	swing calf

Robot actuation occurs via a combination of position and torque control. We command torque control at the hips with PD gains tunable by our algorithm. We commands positions at the knees using a low level controller to regulate swing foot clearance along the terrain. Step size and swing leg step length are also tunable by our algorithm.

Our state measurements consist of the following: i) We used the IMU, gyroscope, and position/velocity feedback resident on every Hebi module. ii) The feet contain sensors developed by the CMU Biorobotics Lab to detect when feet were in contact with the ground. iii) We combined the Realsense cameras D435 and T265 for vision, but had difficulty filtering the information due to the discrete impulses at contact. In the end, the vision measurements are simulated as the robot advances to emulate building a map in real-time using EKF SLAM. All instrumentation is on-board the robot as there is no augmented instrumentation on the boom or additional indoor positioning system. Our controller runs an Extended Kalman Filter (EKF) to obtain state feedback.

Our robot planning and control architecture is shown in Fig. 7.7. Three parallel processes are present and all messages pass via a Robot Operating System (ROS) network. i) Using vision measurements, the terrain map is built using an EKF SLAM algorithm.

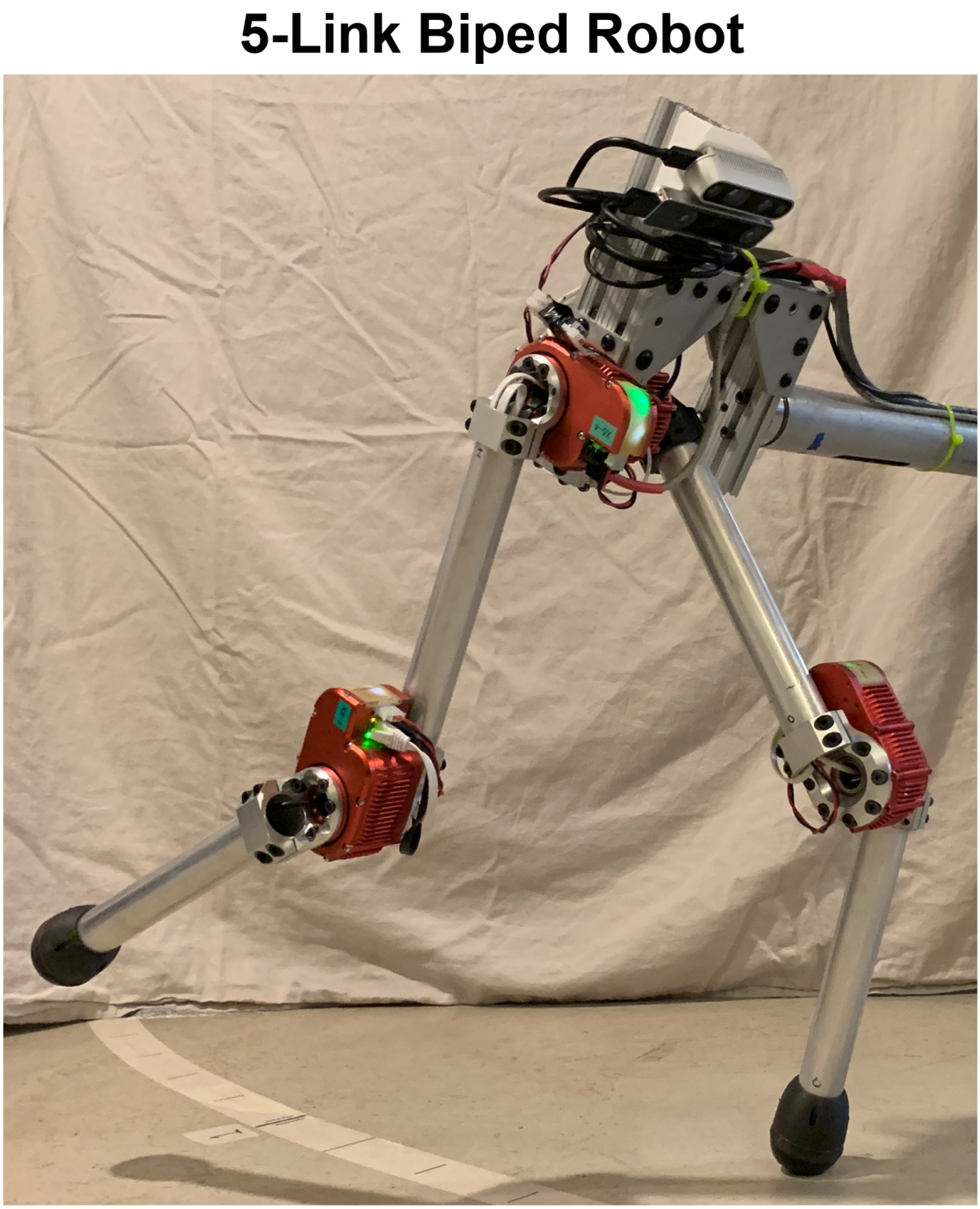

Figure 7.5: 5-Link Biped Robot.

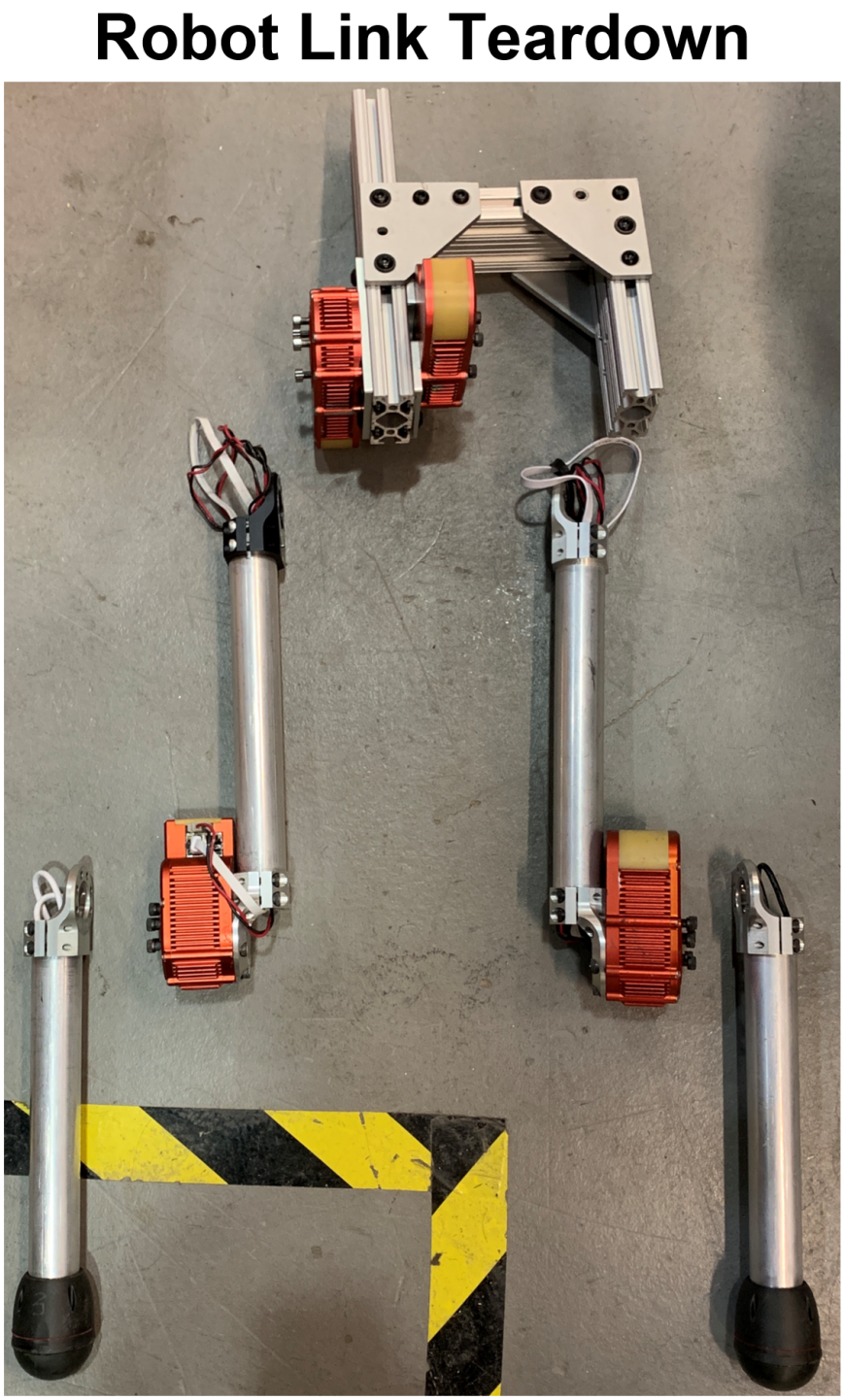

Figure 7.6: Robot Link Teardown. Note, cameras not shown in this image.

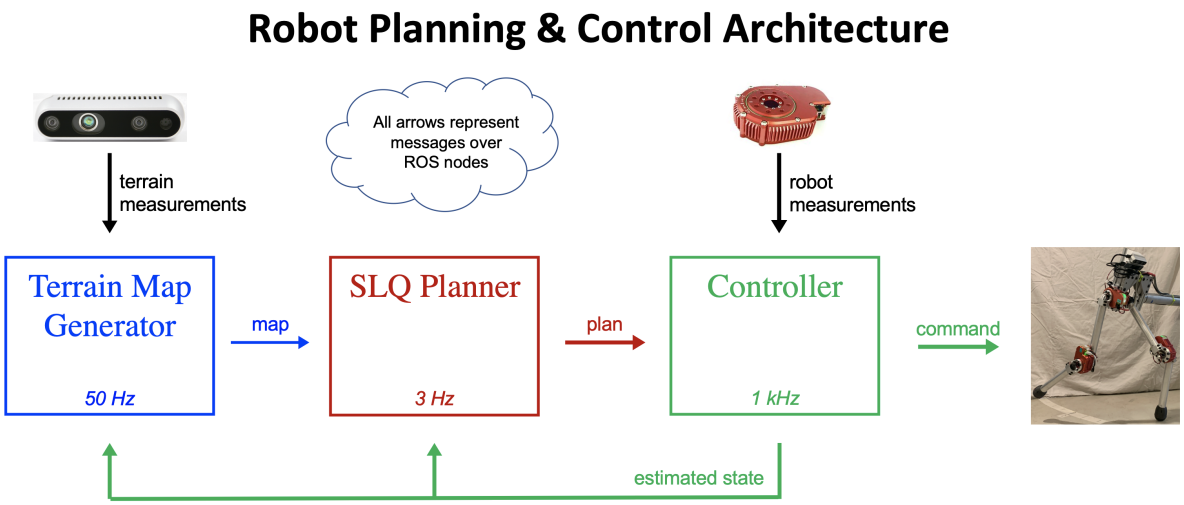

Figure 7.7: Robot Planning and Control Architecture. All boxes represent parallel processes and arrows represent messages passed over Robot Operating System (ROS) network.

ii) The SLQ planner adjusts foothold positions, creates the motion plans, and tunes the controller. iii) Using robot measurements, The controller determines the current state and executes the plan using the prescribed controller.

7.3.2 Hardware Verification

For the hardware configuration, $\mathbf{F}_\mathrm{v}$ is added to (3.4) to represent the viscous friction within the joints

$$\mathbf{M}(\boldsymbol{\Theta})\ddot{\boldsymbol{\Theta}} + \mathbf{C}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})\dot{\boldsymbol{\Theta}} + \mathbf{F}_\mathrm{v}\dot{\boldsymbol{\Theta}} + \mathbf{g}(\boldsymbol{\Theta}) = \boldsymbol{\tau},$$

where $\mathbf{F}_\mathrm{v}$ is a diagonal $S \times S$ matrix.

To demonstrate the planning and execution cycle, we built a 2D course for our robot to navigate, shown in Fig. 7.8. We added two boxes along the path that the robot would require stepping onto or over. We placed the boxes outside of the robot's initial vision, such that the robot could not begin planning for them until it began walking and observed them. At first, the robot begins walking at a repetitive gait over flat terrain. During the 4th step (a), the robot senses the an obstacle and updates the terrain map. At the beginning of the 5th step (b), the planner ingests the newest terrain map, loaded with the

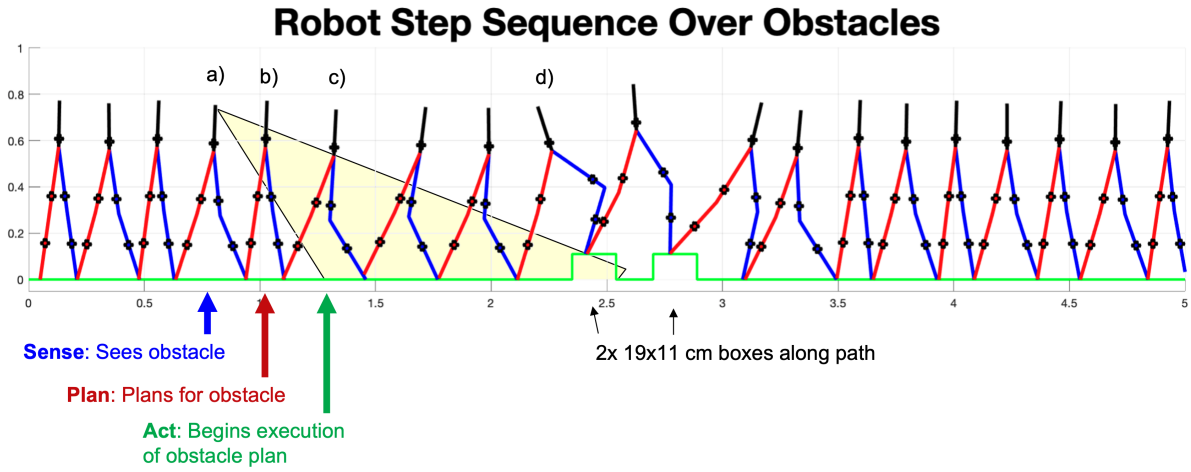

Figure 7.8: Robot Steps Sequence Over Obstacles. The robot steps are shown over the terrain truth in this figure, but planning occurred over a map derived from EKF SLAM. The robot begins by walking over with some nominal gait over the flat terrain. a) On step 4, the robot observes an obstacle, including it in the terrain map. b) On step 5, the robot loads the most recent terrain map, containing the obstacle and begins to plan a sequence for the obstacle. Additionally, the robot senses additional obstacle information and updates the terrain map. c) On step 6, the robot begins to execute the obstacle plan by taking more aggressive steps to gain energy into the system. The system meanwhile continues to refine the plan and adds the second obstacle to the terrain map. d) On step 9, the robot executes a step up onto the obstacle, then across and down. A video demonstrating these results is available at this link.

obstacle. The energy technique is used to seed the initial step up onto the obstacle and then the iterative optimization begins. The planner adjusts the gait by planning more aggressive stepping to add energy to the system. Meanwhile the robot is sensing the gap. During the 6th step (b), the robot begins its first execution of the plan, using the controller designed for this particular step. The step is far more aggressive than previous steps and the knee is bent at impact. This bent knee actually acts as a lift-off during the subsequent steps 7. By the 9th step (d), the robot has enough information and energy to step up and over the obstacles.

The robot walks with an apparent bifurcation by observing Fig 7.8 without context. In order to restrict the robot to a plane, we attached the robot to a boom, thus creating an arc on which the robot walks, shown in Fig. 7.9. We defined the terrain by the center-line of the arc. Therefore, the outer leg travels farther when the inner leg is the stance leg and pivot.

We instructed the robot to use a knee-backward bend during normal flat walking as

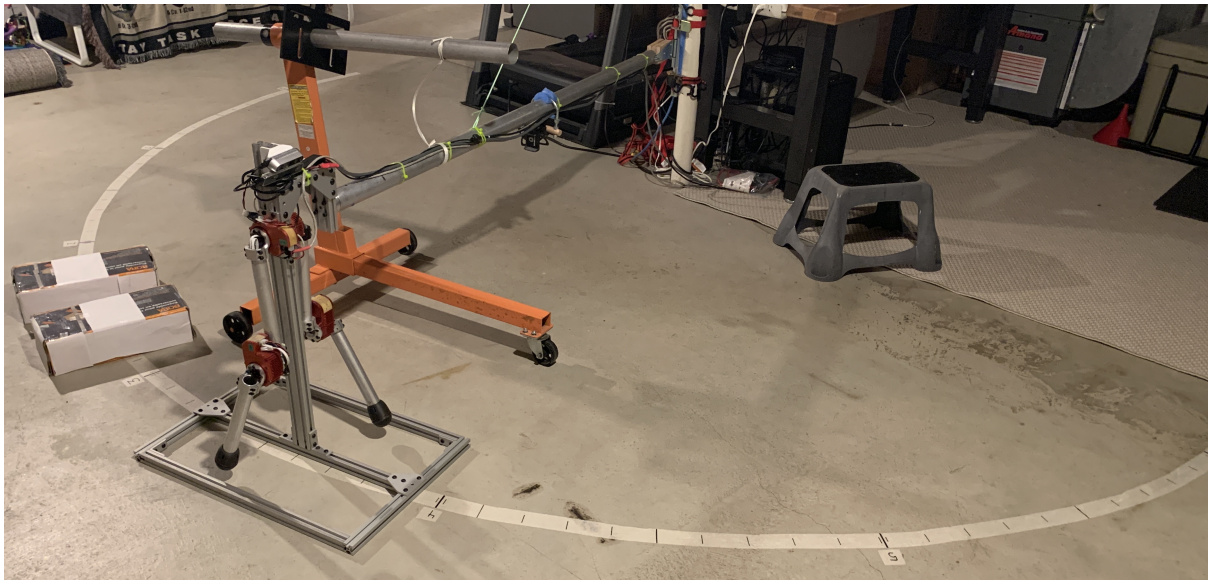

Figure 7.9: Robot Restricted to a Plane, following an Arc. The robot follows an arc, in which the center-line represents the terrain in 2D. The obstacles from Fig. 7.8 can be seen in the image. The robot is resting on a stand that is removed when walking. Also shown is the safety mechanism that that provides no support, but catches the robot if it falls.

we found it far more stable during testing. Yet when applying this technique to terrain, we found that the foot can rarely clear the obstacles fast enough. Therefore, we told the robot to use a forward knee-bend when walking up and down terrain as it was easier to swing the knee forward while lifting the trailing foot over obstacles, especially corners. This knee-bending can be seen in Fig. 7.8.

For terrain cost, we assigned high cost to stepping near corners for a few reason. Firstly, corners are hard to clear when stepping up terrain. Secondly, the robot could slip off of a high corner. And finally, our uncertainty as to exactly where we were on the map was high, so corners were deemed as dangerous.

Additionally, we assigned higher cost to changing heights. Therefore, it could be cheaper to step over an obstacle than to step upon it, but this also means clearing staying away from the corners. Also, it is hard to execute a plan to step over an obstacle, when we might not have been able to see the other side of the obstacle two steps back in the "sense" phase.

We encountered position error (see Fig. 7.10) from a few sources. First and foremost, we lacked absolute position sensing. We relied solely on the information from the onboard

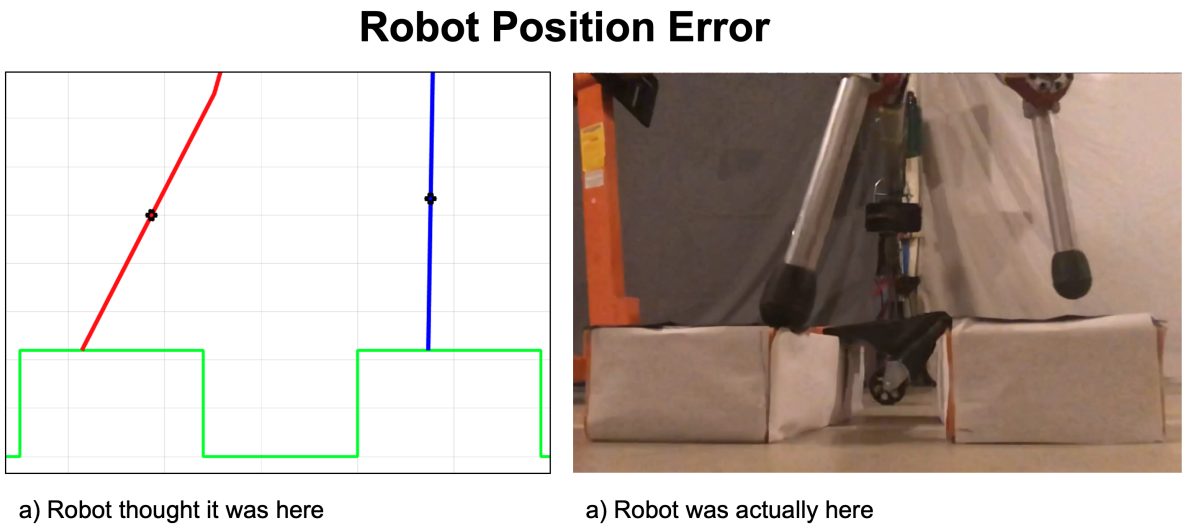

Figure 7.10: Robot Position Error.

sensors, such as accelerometers and gyros. Many of the measurements went wild at discrete impact, thus we had to rely more on kinematic information and internal joint angles to determine how far we stepped. Second, the difference between inner and outer steps due to walking along an arc created. Since we measured from the center-line of the arc, the outside leg (shown in stance in Fig. 7.10) always stepped further than expected. Third, our camera was not a true camera. We used a true [center-line] map and created our 2D terrain using a projection of the camera. Therefore, positioning could not be corrected for using our EKF SLAM algorithm.

Finally, the following shows our representation on our projection mapping:

$$\mathbf{u}_{[\mathrm{n}]} = \begin{bmatrix} K_{p_3} \\ K_{p_4} \\ K_{d_3} \\ K_{d_4} \end{bmatrix} \tag{7.1}$$

$$\mathbf{x}_{[\mathrm{n}]} = \begin{bmatrix} l_{step} \\ l_{swing} \\ E_{[n]} \end{bmatrix}. \tag{7.2}$$

The controls are PD controls over the hips. Since the hips did most of the nonconservative

work for a step, we gave hip control to the trajectory optimization framework. Conversely, we used an internal algorithm for knee control as it was easier for foot clearance of obstacles. Additionally, the calves were of low mass and inertia, therefore the system would not have gained much by utilizing them.

While the system did not have control over the knee torques, it did have control over the swing leg length at touchdown. l_{swing} represents the target length for the swing leg on it's next contact, shown in Fig. 7.11. Since we already specified the direction of the knee bending (negative or positive), the algorithm could only tweek the length from 0.1-1.0 of normal length. Additionally, the robot had control over the length of the step l_{step}, which corresponds to the direct distance between feet rather than the transverse distance along the terrain. We specified the stance leg should be straight at next impact, as well as the hip should bisect the feet. By combining l_{swing} and l_{step}, the algorithm had the rest of the control of the configuration. This configuration is shown in Fig. 7.11. Additionally, since we used PD control on all joints, the internal configuration should ideally not be moving at impact, and all velocity is that about the stance foot. Therefore, we applied a relationship between the potential energy of the configuration and the kinetic energy earned from rotating about the stance foot, giving the robot total energy $E_{[\mathrm{n}]}$ as its final state.

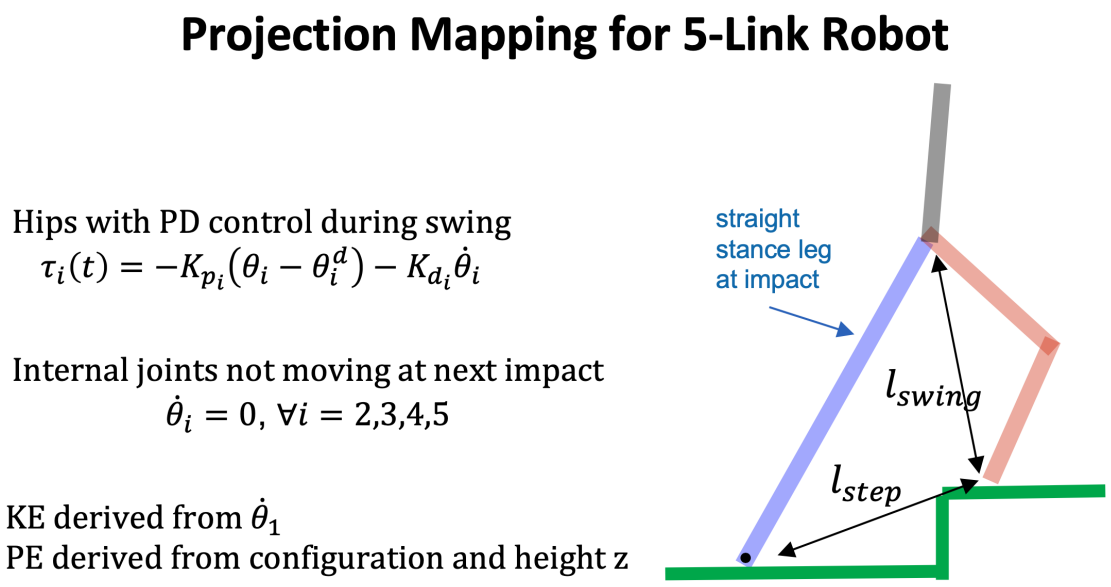

Figure 7.11: Projection Mapping for 5-Link Robot. Using the estimated terrain, the entire configuration can be defined for the configuration using l_{step} and l_{swing}. This assumes we know the swing knee bend direction, the stance knee if fixed straight, and the hip bisects the feet.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

By discretizing the system dynamics over a single step and imposing soft penalties on footstep locations directly into the cost function, this work demonstrates how footstep location, the motion plan, and feedback control can all be implicitly planned using a single approach based on SLQ (see Fig 1.1). One of the main reasons we think that prior employments similar to this are so brittle is that they treat this contact as if it doesn't exist or as a separate event isolated from the system dynamics. For this reason, our unified algorithm combines the continuous dynamics of a step and discrete impact into one hybrid event. Through a linearization of our trajectory once-per-step at the moment of foot contact, our trajectory optimization places the foot along the terrain. We then project it onto a lower dimensional *s.o.s.* for optimization. Energy-based methods give rise to regions for safe-stepping, offering a simple way to seed initial trajectories for seemingly complex dynamical systems. The method was verified with several dynamic models and validated on robotic hardware.

8.2 Future Work

The most clear direction would be to explore different parameterizations of the optimization space. One possible method would be to utilize the coefficients c_i of polynomial

trajectories, i.e. $x(t) = c_n x(t)^n + \ldots + c_1 x(t) + c_0$, where the parameters c_i are controlled by the SLQ optimization. By tuning c_i for each step, the shape of the trajectories could be modified. Another possible parameterization would be to explore transverse linearization [100], where an exponentially-stabilizing controller is built specifically to pull the system along a moving *s.o.s.* transverse to the course of motion. While this method has been used for stabilizing multi-DOF underactuated walking systems [101], it should prove beneficial to bake the linearization into the dynamic controller of the continuous dynamics, resulting tailored region of attraction for each step.

Another way forward would be to confine the foothold position behind the *curve of capture* (see Chap. 5) by the use of constrained optimization. The curve could be established as a state constraint on the *s.o.s.* (see Chap. 4), containing the foothold positions safely within the walking region.

This method could be applied to other systems with hybrid dynamics outside of robotics and engineering. Examples include biological systems in which neurons have different thresholds for firing, or switched systems theory.

8.3 Open Problems

We demonstrated that the unified planning framework can alter a gait through the use of trajectory optimization, but we did not address the stability of the controller. We think the best way forward is using sum-of-squares (SOS) programming to determine the basins of attraction for the control sequence. Using SOS analysis, we would determine that the mouth of one stable control funnel leads into the pre-image of the next funnel in a method similar to the LQR trees developed in [102], [103]. A "funnel" in this context is a feedback policy that collapses a large set of initial conditions into a smaller set of final conditions as shown in Fig. 8.1. If this is true for all N steps for the control sequence, the system can be determined as stable for some calculable deviation around the nominal trajectory. Since we use a quadratic form for the cost-to-go function shown in App. A, we propose that the SLQ value function is our Lyapunov candidate.

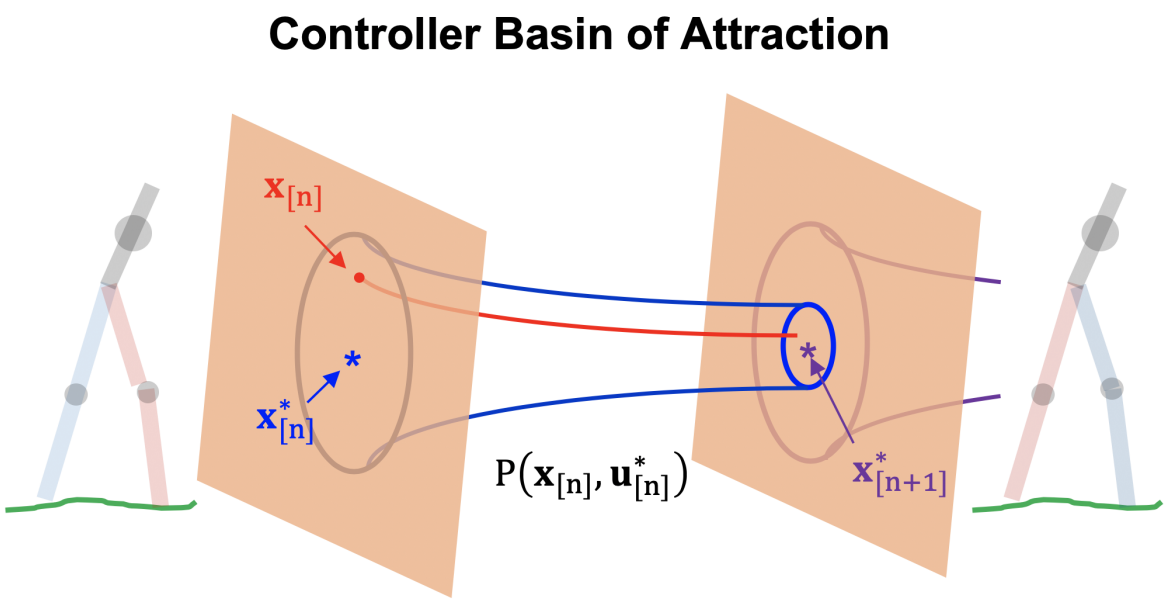

Figure 8.1: Controller Basin of Attraction. For a finite-horizon problem with N individually-tailored controllers, the system is stable if state $\mathbf{x}_{[n]}$ starts within the mouth of a control funnel that leads to the mouth of the next funnel. The shape of the funnel comes from the Lyapunov function.

[104], [105] showed that SOS programming can work in real-time and also [106] demonstrated that LQR trees can be computed online. Therefore, it should be feasible to compute the funnels around the finite-horizon nominal step sequence in real-time. Since we only truly need to re-plan when the state approaches the boundary of the current controller, we could use this information to alter our MPC horizon by embedding SOS into our planner.

Another important aspect to analyze is the effect of the discretization on the stability of the system. The most direct comparison is to that of is that to classically-applied SLQ as shown in Fig. 8.2. This discretization disparity becomes a direct trade-off between control authority and computational speed. On the upper end, the hybrid trajectory is discretized into dozens if not hundreds of segments S. In our unified architecture, we are solving fewer discrete points, giving us fewer decision variables about how to apply control over a particular window of time.

Having less control decisions could offer problems controlling higher order humanoid systems over very complex terrain. Therefore, we propose that the discretization is

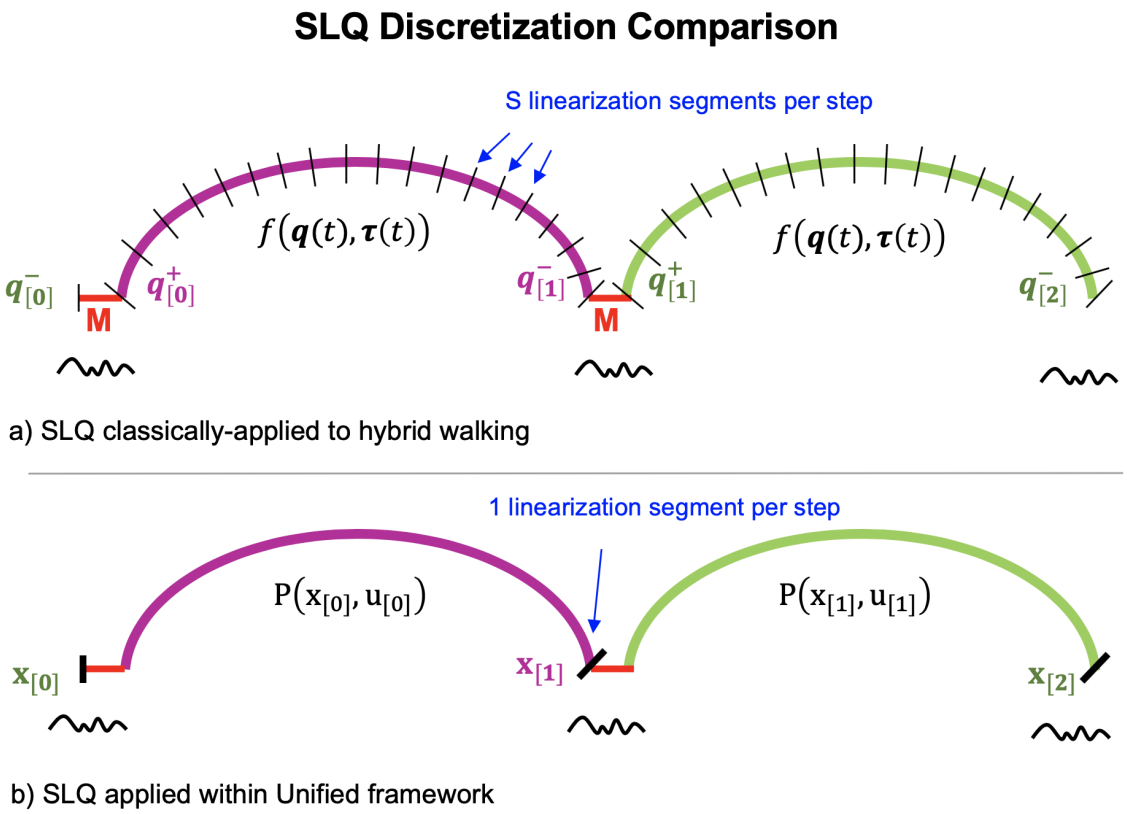

Figure 8.2: Sequential Linear Quadratic Discretization Comparison. a) When classically-applied, the continuous phase is finely discretized. Combined with the discrete impact map, there are a total of S segments per step. Over the course of N steps, there are NS segments over which trajectory optimization and control updates must occur. b) During our unified planning approach, we linearize the entire hybrid dynamics at once, leading to a single segment per step. The trajectory optimization occurs at only N segments.

analyzed in more detail. Begin with classically-applied SLQ in which the segments are very finely applied to generate a baseline for performance. Slowly increase the size of those segments until we only have one segment per the unified architecture. We must determine at what point, if any, control authority is lost or terrain becomes impassible. Additionally, the unified architecture offers footstep placement as a byproduct of the linearization. It must also be quantified as to how this linearization actually assists walking over rough terrain.

References

[1] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain", in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 811–818.

[2] M. A. Arain, I. Havoutis, C. Semini, J. Buchli, and D. G. Caldwell, "A comparison of search-based planners for a legged robot", in *9th International Workshop on Robot Motion and Control*, IEEE, 2013, pp. 104–109.

[3] A. Winkler, I. Havoutis, S. Bazeille, J. Ortiz, M. Focchi, R. Dillmann, D. Caldwell, and C. Semini, "Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 6476–6482.

[4] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 5148–5154.

[5] C. Mastalli, I. Havoutis, A. W. Winkler, D. G. Caldwell, and C. Semini, "On-line and on-board planning and perception for quadrupedal locomotion", in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TEPRA)*, IEEE, 2015, pp. 1–7.

[6] D. Belter, P. Łabecki, and P. Skrzypczyński, "Adaptive motion planning for autonomous rough terrain traversal with a walking robot", *Journal of Field Robotics*, vol. 33, no. 3, pp. 337–370, 2016.

[7] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

[8] K. Hauser, T. Bretl, J.-C. Latombe, and B. Wilcox, "Motion planning for a six-legged lunar robot", in *Algorithmic Foundation of Robotics VII*, Springer, 2008, pp. 301–316.

[9] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain", *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[10] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "An adaptive action model for legged navigation planning", in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2007, pp. 196–202.

[11] M. Vukobratovic and D. Juricic, "Contribution to the synthesis of biped gait", *IEEE Transactions on Biomedical Engineering*, no. 1, pp. 1–6, 1969.

[12] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid", in *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, 2005, pp. 629–634.

[13] H. Liu and Q. Sun, "Motion Planning for Humanoid Robots in Complex Environments Based on Stance Sequence and ZMP Reference", in *Applied Mechanics and Materials*, Trans Tech Publ, vol. 197, 2012, pp. 415–422.

[14] M. Vukobratović and B. Borovac, "Zero-moment point—thirty five years of its life", *International journal of humanoid robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[15] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic, *Biped locomotion: dynamics, stability, control and application*. Springer Science & Business Media, 2012, vol. 7.

[16] R. D. Gregg, A. K. Tilton, S. Candido, T. Bretl, and M. W. Spong, "Control and planning of 3-D dynamic walking with asymptotically stable gait primitives", *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1415–1423, 2012.

[17] I. R. Manchester and J. Umenberger, "Real-time planning with primitives for dynamic walking over uneven terrain", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 4639–4646.

[18] H. Dai and R. Tedrake, "Optimizing robust limit cycles for legged locomotion on unknown terrain", in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, IEEE, 2012, pp. 1207–1213.

[19] B. BAceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2017.

[20] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and foothold optimization for quadruped locomotion", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5308–5313.

[21] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[22] *Motion planning for legged and humanoid robots, author=Hauser, Kris*. Citeseer, 2008.

[23] C. Eldershaw and M. Yim, "Motion planning of legged vehicles in an unstructured environment", in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, IEEE, vol. 4, 2001, pp. 3383–3389.

[24] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, *et al.*, "No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge", in *2015 IEEE-RAS 15th*

International Conference on Humanoid Robots (Humanoids), IEEE, 2015, pp. 623–630.

[25] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, "Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals", in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2015, pp. 1028–1035.

[26] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge", *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, 2015.

[27] G. Wu and K. Sreenath, "Safety-critical and constrained geometric control synthesis using control lyapunov and control barrier functions for systems evolving on manifolds", in *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 2038–2044.

[28] Q. Nguyen and K. Sreenath, "Safety-critical control for dynamical bipedal walking with precise footstep placement", *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 147–154, 2015.

[29] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. Westervelt, C. C. De Wit, and J. Grizzle, "Rabbit: A testbed for advanced control theory", 2003.

[30] Q. Nguyen, X. Da, J. Grizzle, and K. Sreenath, "Dynamic walking on stepping stones with gait library and control barrier functions", *Arbor*, vol. 1001, p. 48 109, 2016.

[31] A. Ramezani and J. W. Grizzle, "ATRIAS 2.0, a new 3D bipedal robotic walker and runner", in *Adaptive Mobile Robotics*, World Scientific, 2012, pp. 467–474.

[32] M. Zucker, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "An optimization approach to rough terrain locomotion", in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 3589–3595.

[33] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning", *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[34] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact", in *Algorithmic foundations of robotics X*, Springer, 2013, pp. 527–542.

[35] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact", *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[36] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization", *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[37] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems", *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[38] A. Sideris and L. A. Rodriguez, "A riccati approach to equality constrained linear quadratic optimal control", in *Proceedings of the 2010 American Control Conference*, IEEE, 2010, pp. 5167–5172.

[39] J. van den Berg, "Iterated LQR smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost", in *2014 American Control Conference*, IEEE, 2014, pp. 1912–1918.

[40] M. Giftthaler, F. Farshidian, T. Sandy, L. Stadelmann, and J. Buchli, "Efficient kinematic planning for mobile manipulators with non-holonomic constraints using optimal control", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 3411–3417.

[41] F. Farshidian, M. Kamgarpour, D. Pardo, and J. Buchli, "Sequential linear quadratic optimal control for nonlinear switched systems", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1463–1469, 2017.

[42] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 5674–5680.

[43] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous Driving Motion Planning with Constrained Iterative LQR", *IEEE Transactions on Intelligent Vehicles*, 2019.

[44] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 93–100.

[45] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.", in *ICINCO (1)*, 2004, pp. 222–229.

[46] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems", in *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, 2005, pp. 300–306.

[47] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking", in *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2016, pp. 1398–1404.

[48] S. Crews, S. Agrawal, and M. Travers, "Unified Foothold Selection and Motion Planning for Legged Systems in Real-Time", in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2019, pp. 622–629.

[49] S.-C. Chang, C.-H. Chen, I.-K. Fong, and P. B. Luh, "Hydroelectric generation scheduling with an effective differential dynamic programming algorithm", *IEEE transactions on power systems*, vol. 5, no. 3, pp. 737–743, 1990.

[50] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 1168–1175.

[51] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 695–702.

[52] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery", in *2006 6th IEEE-RAS international conference on humanoid robots*, IEEE, 2006, pp. 200–207.

[53] T. McGeer, "Stability and control of two-dimensional biped walking", *Center for Systems Science, Simon Fraser University, Burnaby, BC, Canada, Technical Report*, vol. 1, 1988.

[54] T. McGeer *et al.*, "Passive dynamic walking", *I. J. Robotic Res.*, vol. 9, no. 2, pp. 62–82, 1990.

[55] A. Goswami, B. Espiau, and A. Keramane, "Limit cycles in a passive compass gait biped and passivity-mimicking control laws", *Autonomous Robots*, vol. 4, no. 3, pp. 273–286, 1997.

[56] B. Thuilot, A. Goswami, and B. Espiau, "Bifurcation and chaos in a simple passive bipedal gait", in *Proceedings of International Conference on Robotics and Automation*, IEEE, vol. 1, 1997, pp. 792–798.

[57] M. W. Spong *et al.*, "Passivity based control of the compass gait biped", in *IFAC world congress*, Citeseer Beijing, China, vol. 3, 1999, pp. 19–23.

[58] A. D. Kuo, "Stabilization of lateral motion in passive dynamic walking", *The International journal of robotics research*, vol. 18, no. 9, pp. 917–930, 1999.

[59] K. Osuka and K.-I. Kirihara, "Motion analysis and experiment of passive walking robot Quartet II", *Journal of the Robotics Society of Japan*, vol. 18, no. 5, pp. 737–742, 2000.

[60] M. W. Spong and G. Bhatia, "Further results on control of the compass gait biped", in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, IEEE, vol. 2, 2003, pp. 1933–1938.

[61] F. Asano, "Stability analysis of passive compass gait using linearized model", in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 557–562.

[62] H. Gritli, N. Khraeif, and S. Belghith, "Falling of a passive compass-gait biped robot caused by a boundary crisis", in *Proceedings of the 4th Chaotic Modeling and Simulation International Conference, Crete, Greece*, 2011, pp. 155–162.

[63] J. Rebula, F. Canas, J. Pratt, and A. Goswami, "Learning capture points for humanoid push recovery", in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2007, pp. 65–72.

[64] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models", *The international journal of robotics research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[65] M. Krause, J. Englsberger, P.-B. Wieber, and C. Ott, "Stabilization of the capture point dynamics for bipedal walking based on model predictive control", *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 165–171, 2012.

[66] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online walking pattern generation for push recovery and minimum delay to commanded change of direction and speed", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 3411–3416.

[67] M. Shafiee-Ashtiani, A. Yousefi-Koma, M. Shariat-Panahi, and M. Khadiv, "Push recovery of a humanoid robot based on model predictive control and capture point", in *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, IEEE, 2016, pp. 433–438.

[68] M.-y. Deng, Z.-y. Ma, Y.-n. Wang, H.-s. Wang, Y.-b. Zhao, Q.-x. Wei, W. Yang, and C.-j. Yang, "Fall preventive gait trajectory planning of a lower limb rehabilitation exoskeleton based on capture point theory", *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 10, pp. 1322–1330, 2019.

[69] M. Missura, M. Bennewitz, and S. Behnke, "Capture steps: Robust walking for humanoid robots", *International Journal of Humanoid Robotics*, p. 1 950 032, 2020.

[70] S. Mochon and T. A. McMahon, "Ballistic walking: An improved model", *Mathematical Biosciences*, vol. 52, no. 3-4, pp. 241–260, 1980.

[71] T. Anstensrud, "2-d passive compass biped walker", PhD thesis, Master's thesis, NTNU, 2013.

[72] P. A. Bhounsule, "Control of a compass gait walker based on energy regulation using ankle push-off and foot placement", *Robotica*, vol. 33, no. 6, pp. 1314–1324, 2015.

[73] A. J. McCaffery, Z. T. Alwahabi, M. A. Osborne, and C. J. Williams, "Rotational transfer, an angular momentum model", *The Journal of chemical physics*, vol. 98, no. 6, pp. 4586–4602, 1993.

[74] M. W. Spong and F. Bullo, "Controlled symmetries and passive walking", in *Proceeding of The 15th Triennial World Congress*, Barcelona, Spain, 2002.

[75] R. Tedrake, "Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines: Course Notes for MIT 6.832", *Working draft edition*, vol. 3, 2009.

[76] Y. Zhao, B. R. Fernandez, and L. Sentis, "Robust Phase-Space Planning for Agile Legged Locomotion over Various Terrain Topologies.", in *Robotics: Science and Systems*, 2016.

[77] V. Janardhan and R. P. Kumar, "Online trajectory generation for wide ditch crossing of biped robots using control constraints", *Robotics and Autonomous Systems*, vol. 97, pp. 61–82, 2017.

[78] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2006.

[79] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[80] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices", 1955.

[81] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation", in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, IEEE, vol. 1, 2001, pp. 239–246.

[82] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Resolved momentum control: Humanoid motion planning based on the linear and angular momentum", in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, IEEE, vol. 2, 2003, pp. 1644–1650.

[83] I.-W. Park, J.-Y. Kim, J. Lee, and J.-H. Oh, "Online free walking trajectory generation for biped humanoid robot KHR-3 (HUBO)", in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 1231–1236.

[84] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics", in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 4420–4427.

[85] J. J. Alcaraz-Jiménez, D. Herrero-Pérez, and H. Martınez-Barberá, "Robust feedback control of ZMP-based gait for the humanoid robot Nao", *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1074–1088, 2013.

[86] F. Ferro and L. Marchionni, "Reem: A humanoid service robot", in *ROBOT2013: First Iberian Robotics Conference*, Springer, 2014, pp. 521–525.

[87] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, *et al.*, "Overview of the torque-controlled humanoid robot TORO", in *2014 IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2014, pp. 916–923.

[88] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, *et al.*, "TALOS: A new humanoid research platform targeted for industrial applications", in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, IEEE, 2017, pp. 689–695.

[89] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based zmp constraints", *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2201–2208, 2017.

[90] K. Byl and R. Tedrake, "Approximate optimal control of the compass gait on rough terrain", in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 1258–1263.

[91] T. Pappas, A. Laub, and N. Sandell, "On the numerical solution of the discrete-time algebraic Riccati equation", *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 631–641, 1980.

[92] D. Noid and R. Marcus, "Semiclassical calculation of bound states in a multidimensional system. Use of Poincaré's surface of section", *The Journal of Chemical Physics*, vol. 62, no. 6, pp. 2119–2124, 1975.

[93] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems", in *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, 2005, pp. 2275–2280.

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

[95] H. R. Vejdani, A. Wu, H. Geyer, and J. W. Hurst, "Touch-down angle control for spring-mass walking", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 5101–5106.

[96] O. E. Ramos and K. Hauser, "Generalizations of the capture point to nonlinear center of mass paths and uneven terrain", in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2015, pp. 851–858.

[97] K. Ochs, "A comprehensive analytical solution of the nonlinear pendulum", *European Journal of Physics*, vol. 32, no. 2, p. 479, 2011.

[98] E. Ackerman, "Agility Robotics Introduces Cassie, a Dynamic and Talented Robot Delivery Ostrich", *IEEE Spectrum (Automation Blog). Abgerufen am*, vol. 28, p. 2018, 2017.

[99] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, "Fast online trajectory optimization for the bipedal robot cassie", in *Robotics: Science and Systems*, 2018.

[100] G. A. Leonov, "Generalization of the Andronov Vitt theorem", *Regular and chaotic dynamics*, vol. 11, no. 2, pp. 281–289, 2006.

[101] L. B. Freidovich, A. S. Shiriaev, and I. R. Manchester, "Stability analysis and control design for an underactuated walking robot via computation of a transverse linearization", *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 166–10 171, 2008.

[102] R. Tedrake, "LQR-Trees: Feedback motion planning on sparse randomized trees", 2009.

[103] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification", *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[104] A. A. Ahmadi and A. Majumdar, "Some applications of polynomial optimization in operations research and real-time decision making", *Optimization Letters*, vol. 10, no. 4, pp. 709–729, 2016.

[105] S. Singh, M. Chen, S. L. Herbert, C. J. Tomlin, and M. Pavone, "Robust tracking with model mismatch for fast and safe planning: an SOS optimization approach", *arXiv preprint arXiv:1808.00649*, 2018.

[106] R. Natarajan, S. Rajasekaran, and J. D. Taylor, "Towards Planning and Control of Hybrid Systems with Limit Cycle using LQR Trees", *arXiv preprint arXiv:1711.04063*, 2017.

Appendix A

Planning with a Sequential Linear Quadratic Regulator

In this Appendix, we derive the SLQ used within this work.

A.0.1 Problem Definition

$$\texttt{Minimize} \qquad J[\mathbf{x}(\cdot), \mathbf{u}(\cdot)] = \alpha^{N}\Phi(\mathbf{x}_{[\mathrm{n}]}) + \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \alpha^{\mathrm{k}}\mathrm{L}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})$$

$$\texttt{Subject to} \qquad \mathbf{x}_{[\mathrm{n}+1]} = P(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) \qquad n \in \{0, 1, \ldots, N-1\}$$

$$\mathbf{x}_{[\mathrm{n}]} = \mathbf{x}(\mathrm{t}_{[\mathrm{n}]}) \qquad n \in \{0, 1, \ldots, N\}$$

n	`discrete time index`
$\mathbf{x}_{[\mathrm{n}]}$	`system state at segment` n
$\mathbf{u}_{[\mathrm{n}]}$	`control input at segment` n
$P(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})$	`state transition equation`
α	`discount factor/decay rate` $0 \leq \alpha \leq 1$

For finite horizon case, discount factor α is usually set to 1

$$J = \Phi(\mathbf{x}_{[\mathrm{n}]}) + \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{L}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})$$

The optimal control policy $\mu^\star$ minimizes the cost function and gives optimal cost $J^\star$

$$\mu^\star = \arg\min_{\mathbf{u}} J[\mathbf{x}(\cdot), \mathbf{u}(\cdot)]$$

A.0.2 Equations of Motion

Initial stable control policy $\mu(\mathrm{n}, \mathbf{x})$ gives us control input trajectory $\bar{\mathbf{x}}_{[\mathrm{n}]}$ and nominal state-trajectory $\bar{\mathbf{u}}_{[\mathrm{n}]}$. We will linearize about every state-action pair $(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})$.

$$\bar{\mathbf{x}}_{[\mathrm{n}]} \triangleq \mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^*_{[\mathrm{n}]} \qquad \texttt{state increment}$$

$$\bar{\mathbf{u}}_{[\mathrm{n}]} \triangleq \mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^*_{[\mathrm{n}]} \qquad \texttt{control input increment}$$

Approximate $P(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})$ by its first order Taylor expansion about fixed point $(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})$.

$$\begin{aligned}\mathbf{x}_{[\mathrm{n}+1]} &= P(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) \\ &\approx P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}) + \frac{\partial P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})}{\partial \mathbf{x}} \bar{\mathbf{x}}_{[\mathrm{n}]} + \frac{\partial P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})}{\partial \mathbf{u}} \bar{\mathbf{u}}_{[\mathrm{n}]}\end{aligned}$$

By setting $\mathbf{x}^*_{[\mathrm{n+1}]} = P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})$, we obtain

$$\begin{aligned}
\bar{\mathbf{x}}_{[\mathrm{n+1}]} &\approx \frac{\partial P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})}{\partial \mathbf{x}} \bar{\mathbf{x}}_{[\mathrm{n}]} + \frac{\partial P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})}{\partial \mathbf{u}} \bar{\mathbf{u}}_{[\mathrm{n}]} \\
&\approx \mathbf{A}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]} + \mathbf{B}_{[\mathrm{n}]} \bar{\mathbf{u}}_{[\mathrm{n}]} \\
\mathbf{A}_{[\mathrm{n}]} &= \frac{\partial P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})}{\partial \mathbf{x}} && (N_x \times N_x) \ \texttt{matrix} \\
\mathbf{B}_{[\mathrm{n}]} &= \frac{\partial P(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})}{\partial \mathbf{u}} && (N_x \times N_u) \ \texttt{matrix}
\end{aligned}$$

A.0.3 Cost Function

$$\begin{aligned}
J =& \Phi(\mathbf{x}_{[\mathrm{N}]}) + \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{L}_{[\mathrm{n}]}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) \\
=& (\mathbf{x}_{[\mathrm{N}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{N}]})^{\mathrm{T}} \mathbf{Q}_{[\mathrm{N}]} (\mathbf{x}_{[\mathrm{N}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{N}]}) + p_{[\mathrm{N}]}(\mathbf{x}_{[\mathrm{N}]}) \\
&+ \sum_{n=0}^{N-1} \{ (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{n}]})^{\mathrm{T}} \mathbf{Q}_{[\mathrm{n}]} (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{n}]}) + p_{[\mathrm{n}]}(\mathbf{x}_{[\mathrm{n}]}) + (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^{\mathrm{d}}_{[\mathrm{n}]})^{\mathrm{T}} \mathbf{R}_{[\mathrm{n}]} (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^{\mathrm{d}}_{[\mathrm{n}]}) \}
\end{aligned}$$

Approximate J by its second order Taylor expansion.

$$\begin{aligned}
c(x,u) \approx& c(x^a, u^a) + \frac{\partial c(x^a, u^a)}{\partial x} \delta x + \frac{\partial c(x^a, u^a)}{\partial u} \delta u \\
&+ \frac{1}{2} \left(\frac{\partial^2 c(x^a, u^a)}{\partial x^2} (\delta x)^2 + \frac{\partial^2 c(x^a, u^a)}{\partial u^2} (\delta u)^2 + 2 \frac{\partial}{\partial u} \left(\frac{\partial c(x^a, u^a)}{\partial x} \delta x \right) \delta u \right)
\end{aligned}$$

For n = N:

$$\begin{aligned}
\Phi(\mathbf{x}_{[\mathrm{N}]}) =& (\mathbf{x}_{[\mathrm{N}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{N}]})^{\mathrm{T}} \mathbf{Q}_{[\mathrm{N}]} (\mathbf{x}_{[\mathrm{N}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{N}]}) + p_{[\mathrm{N}]}(\mathbf{x}_{[\mathrm{N}]}) \\
\approx& \Phi^* + \mathbf{\Phi}^*_{\mathrm{x}} (\mathbf{x}_{[\mathrm{N}]} - \mathbf{x}^*_{[\mathrm{N}]}) + \frac{1}{2} (\mathbf{x}_{[\mathrm{N}]} - \mathbf{x}^*_{[\mathrm{N}]})^{\mathrm{T}} \mathbf{\Phi}^*_{xx} (\mathbf{x}_{[\mathrm{N}]} - \mathbf{x}^*_{[\mathrm{N}]}) \\
\approx& \Phi^* + \mathbf{\Phi}^*_{\mathrm{x}} \bar{\mathbf{x}}_{[\mathrm{N}]} + \frac{1}{2} \bar{\mathbf{x}}^{\mathrm{T}}_{[\mathrm{N}]} \mathbf{\Phi}^*_{xx} \bar{\mathbf{x}}_{[\mathrm{N}]}
\end{aligned}$$

$$
\begin{aligned}
\Phi^* =& \Phi(\mathbf{x}_{[\mathrm{N}]})\Big|_{\mathbf{x}^*_{[\mathrm{N}]},0} && (1 \times 1) \quad \texttt{scalar} \\
\mathbf{\Phi}^*_{\mathrm{x}} =& \frac{\partial \Phi(\mathbf{x}_{[\mathrm{N}]})}{\partial \mathbf{x}}\Big|_{\mathbf{x}^*_{[\mathrm{N}]},0} && (1 \times N_x) \quad \texttt{vector} \\
\mathbf{\Phi}^*_{xx} =& \frac{\partial^2 \Phi(\mathbf{x}_{[\mathrm{N}]})}{\partial \mathbf{x}^2}\Big|_{\mathbf{x}^*_{[\mathrm{N}]},0} && (N_x \times N_x) \quad \texttt{matrix}
\end{aligned}
$$

$\forall n \in \{0, \dots, N-1\}$:

$$
\begin{aligned}
L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) =& (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{n}]})^{\mathrm{T}} \mathbf{Q}_{[\mathrm{n}]} (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^{\mathrm{d}}_{[\mathrm{n}]}) + \mathrm{h}(\mathbf{x}_{[\mathrm{n}]}) + (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^{\mathrm{d}}_{[\mathrm{n}]})^{\mathrm{T}} \mathbf{R}_{[\mathrm{n}]} (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^{\mathrm{d}}_{[\mathrm{n}]}) \\
\approx& L^* + (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^*_{[\mathrm{n}]})^{\mathrm{T}} \mathbf{L}^*_{xu} (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^*_{[\mathrm{n}]}) \\
&+ \mathbf{L}^*_x (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^*_{[\mathrm{n}]}) + \frac{1}{2} (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^*_{[\mathrm{n}]})^{\mathrm{T}} \mathbf{L}^*_{xx} (\mathbf{x}_{[\mathrm{n}]} - \mathbf{x}^*_{[\mathrm{n}]}) \\
&+ \mathbf{L}^*_u (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^*_{[\mathrm{n}]}) + \frac{1}{2} (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^*_{[\mathrm{n}]})^{\mathrm{T}} \mathbf{L}^*_{uu} (\mathbf{u}_{[\mathrm{n}]} - \mathbf{u}^*_{[\mathrm{n}]}) \\
\approx& L^* + \mathbf{L}^*_x \bar{\mathbf{x}}_{[\mathrm{n}]} + \mathbf{L}^*_u \bar{\mathbf{u}}_{[\mathrm{n}]} + \frac{1}{2} \bar{\mathbf{x}}^{\mathrm{T}}_{[\mathrm{n}]} \mathbf{L}^*_{xx} \\
&+ \frac{1}{2} \bar{\mathbf{u}}^T_{[\mathrm{n}]} \mathbf{L}^*_{uu} \bar{\mathbf{u}}_{[\mathrm{n}]} + \frac{1}{2} \bar{\mathbf{x}}^{\mathrm{T}}_{[\mathrm{n}]} \mathbf{L}^*_{xu} \bar{\mathbf{u}}_{[\mathrm{n}]} + \frac{1}{2} \bar{\mathbf{u}}^{\mathrm{T}}_{[\mathrm{n}]} \mathbf{L}^*_{ux} \bar{\mathbf{x}}_{[\mathrm{n}]}
\end{aligned}
$$

$$
\begin{aligned}
L^* &= L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})\Big|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} && (1 \times 1) \ \texttt{scalar} \\
\mathbf{L}^*_x &= \frac{\partial L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{x}}\Big|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} && (1 \times N_x) \ \texttt{vector} \\
\mathbf{L}^*_u &= \frac{\partial L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{u}}\Big|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} && (1 \times N_u) \ \texttt{vector} \\
\mathbf{L}^*_{xx} &= \frac{\partial^2 L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{x}^2}\Big|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} && (N_x \times N_x) \ \texttt{matrix} \\
\mathbf{L}^*_{uu} &= \frac{\partial^2 L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{u}^2}\Big|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} && (N_u \times N_u) \ \texttt{matrix} \\
\mathbf{L}^*_{xu} &= \frac{\partial}{\partial \mathbf{u}}\left(\frac{\partial L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{x}}\right)\Big|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} && (N_x \times N_u) \ \texttt{matrix} \\
\mathbf{L}^*_{ux} &= \frac{\partial}{\partial \mathbf{x}}\left(\frac{\partial L(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{u}}\right)\Big|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} = (\mathbf{L}^*_{xu})^{\mathrm{T}} && (N_u \times N_x) \ \texttt{matrix}
\end{aligned}
$$

$$
\begin{aligned}
J \approx & \Phi^* + \mathbf{\Phi}^*_{\mathrm{x}} \bar{\mathbf{x}}_{[\mathrm{n}]} + \frac{1}{2} \bar{\mathbf{x}}^{\mathrm{T}}_{[\mathrm{n}]} \mathbf{\Phi}^*_{xx} \bar{\mathbf{x}}_{[\mathrm{n}]} \\
& + \sum_{n=0}^{N-1} \left(\frac{1}{2} \begin{bmatrix} \bar{\mathbf{x}}_{[\mathrm{n}]} \\ \bar{\mathbf{u}}_{[\mathrm{n}]} \end{bmatrix}^T \begin{bmatrix} \mathbf{L}^*_{xx} & \mathbf{L}^*_{ux} \\ \mathbf{L}^*_{xu} & \mathbf{L}^*_{uu} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_{[\mathrm{n}]} \\ \bar{\mathbf{u}}_{[\mathrm{n}]} \end{bmatrix} + \begin{bmatrix} \mathbf{L}^*_x & \mathbf{L}^*_u \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_{[\mathrm{n}]} \\ \bar{\mathbf{u}}_{[\mathrm{n}]} \end{bmatrix} + L^* \right)
\end{aligned}
$$

A.0.4 Value Function

Quadratic Value function of state deviation $\bar{\mathbf{x}}_{[\mathrm{n+1}]}$. Recall $\bar{\mathbf{x}}_{[\mathrm{n+1}]} \approx \mathbf{A}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]} + \mathbf{B}_{[\mathrm{n}]} \bar{\mathbf{u}}_{[\mathrm{n}]}$.

$$
\begin{aligned}
V^*_{[n+1]}(\bar{\mathbf{x}}_{[n+1]}) =& \frac{1}{2}\delta\left(\mathbf{x}^{\mathrm{T}}_{[n+1]}\mathbf{S}_{[n+1]} + \mathbf{s}_{[n+1]}\right)\bar{\mathbf{x}}_{[n+1]} + s_{[n+1]} \\
V^*_{[n+1]}(\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]}) =& \frac{1}{2}[\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]}]^{\mathrm{T}}\mathbf{S}_{[n+1]}[\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]}] \\
&+ \mathbf{s}_{[n+1]}[\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]}] + \mathrm{s}_{[n+1]} \\
=& \frac{1}{2}[\bar{\mathbf{x}}^{T}_{[n]}\mathbf{A}^{\mathrm{T}}_{[n]} + \bar{\mathbf{u}}^{\mathrm{T}}_{[n]}\mathbf{B}^{\mathrm{T}}_{[n]}]\mathbf{S}_{[n+1]}[\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]}] \\
&+ \mathbf{s}_{[n+1]}[\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]}] + \mathrm{s}_{[n+1]} \\
=& \frac{1}{2}\bar{\mathbf{x}}^{T}_{[n]}\mathbf{A}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \frac{1}{2}\bar{\mathbf{x}}^{\mathrm{T}}_{[n]}\mathbf{A}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]} \\
&+ \frac{1}{2}\bar{\mathbf{u}}^{T}_{[n]}\mathbf{B}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \frac{1}{2}\bar{\mathbf{u}}^{\mathrm{T}}_{[n]}\mathbf{B}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]} \\
&+ \mathbf{s}_{[n+1]}\mathbf{A}_{[n]}\bar{\mathbf{x}}_{[n]} + \mathbf{s}_{[n+1]}\mathbf{B}_{[n]}\bar{\mathbf{u}}_{[n]} + \mathrm{s}_{[n+1]}
\end{aligned}
$$

Bellman Equation for Value Function at previous time step n

$$
\begin{aligned}
V^*_{[n]}(\bar{\mathbf{x}}_{[n]}) =& \frac{1}{2}\left(\bar{\mathbf{x}}^{T}_{[n]}\mathbf{S}_{[n]} + \mathbf{s}_{[n]}\right)\bar{\mathbf{x}}_{[n]} + s_{[n]} \\
=& \min_{\mathbf{u}_{[n]}}\left[L_{[n]}(\mathbf{x}_{[n]}, \mathbf{u}_{[n]}) + V^*_{[n+1]}(\bar{\mathbf{x}}_{[n+1]})\right]
\end{aligned}
$$

Substituting Running Cost $L_{[n]}$ and Value Function at $n+1$

$$
\begin{aligned}
V^*_{[n]}(\bar{\mathbf{x}}_{[n]}) = \min_{\mathbf{u}_{[n]}} & L^* + \mathbf{L}^*_x \bar{\mathbf{x}}_{[n]} + \mathbf{L}^*_u \bar{\mathbf{u}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]} \mathbf{L}^*_{xx} \bar{\mathbf{x}}_{[n]} + \frac{1}{2}\bar{\mathbf{u}}^T_{[n]} \mathbf{L}^*_{uu} \bar{\mathbf{u}}_{[n]} + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]} \mathbf{L}^*_{xu} \bar{\mathbf{u}}_{[n]} + \frac{1}{2}\bar{\mathbf{u}}^T_{[n]} \mathbf{L}^*_{ux} \bar{\mathbf{x}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]} \mathbf{A}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{A}_{[n]} \bar{\mathbf{x}}_{[n]} + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]} \mathbf{A}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{B}_{[n]} \bar{\mathbf{u}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{u}}^T_{[n]} \mathbf{B}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{A}_{[n]} \bar{\mathbf{x}}_{[n]} + \frac{1}{2}\bar{\mathbf{u}}^T_{[n]} \mathbf{B}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{B}_{[n]} \bar{\mathbf{u}}_{[n]} \\
& + \mathbf{s}_{[n+1]} \mathbf{A}_{[n]} \bar{\mathbf{x}}_{[n]} + \mathbf{s}_{[n+1]} \mathbf{B}_{[n]} \bar{\mathbf{u}}_{[n]} + s_{[n+1]} \\
= \min_{\mathbf{u}_{[n]}} & L^* + s_{[n+1]} \\
& + \left(\mathbf{L}^*_x + \mathbf{s}_{[n+1]} \mathbf{A}_{[n]}\right) \bar{\mathbf{x}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]} \left(\mathbf{L}^*_{xx} + \mathbf{A}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{A}_{[n]}\right) \bar{\mathbf{x}}_{[n]} \\
& + \left(\mathbf{L}^*_u + \mathbf{s}_{[n+1]} \mathbf{B}_{[n]} + \bar{\mathbf{x}}^T_{[n]} \left(\mathbf{L}^*_{xu} + \mathbf{A}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{B}_{[n]}\right)\right) \bar{\mathbf{u}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{u}}^T_{[n]} \left(\mathbf{L}^*_{uu} + \mathbf{B}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{B}_{[n]}\right) \bar{\mathbf{u}}_{[n]} \\
= \min_{\mathbf{u}_{[n]}} & L^* + s_{[n+1]} \\
& + \left(\mathbf{L}^*_x + \mathbf{s}_{[n+1]} \mathbf{A}_{[n]}\right) \bar{\mathbf{x}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]} \left(\mathbf{L}^*_{xx} + \mathbf{A}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{A}_{[n]}\right) \bar{\mathbf{x}}_{[n]} \\
& + \left(\mathbf{g}^{\mathrm{T}}_{[n]} + \bar{\mathbf{x}}^{\mathrm{T}}_{[n]} \mathbf{G}^{\mathrm{T}}_{[n]}\right) \bar{\mathbf{u}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{u}}^T_{[n]} \mathbf{H}^{\mathrm{T}}_{[n]} \bar{\mathbf{u}}_{[n]}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{g}_{[n]} &= \left(\mathbf{L}^*_u + \mathbf{s}_{[n+1]} \mathbf{B}_{[n]}\right)^T && (N_u \times 1) \quad \texttt{vector} \\
\mathbf{G}_{[n]} &= \left(\mathbf{L}^*_{xu} + \mathbf{A}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{B}_{[n]}\right)^T && (N_u \times N_x) \quad \texttt{matrix} \\
\mathbf{H}_{[n]} &= \left(\mathbf{L}^*_{uu} + \mathbf{B}^{\mathrm{T}}_{[n]} \mathbf{S}_{[n+1]} \mathbf{B}_{[n]}\right)^T && (N_u \times N_u) \quad \texttt{matrix}
\end{aligned}
$$

Finding optimal control when value function gradient wrt control goes to 0

$$
\begin{aligned}
\min_{\bar{\mathbf{u}}_{[\mathrm{n}]}} \left[\left(\mathbf{g}_{[\mathrm{n}]}^{\mathrm{T}} + \bar{\mathbf{x}}_{[\mathrm{n}]}^{\mathrm{T}} \mathbf{G}_{[\mathrm{n}]}^{\mathrm{T}} \right) \bar{\mathbf{u}}_{[\mathrm{n}]} + \frac{1}{2} \bar{\mathbf{u}}_{[\mathrm{n}]}^{T} \mathbf{H}_{[\mathrm{n}]}^{\mathrm{T}} \bar{\mathbf{u}}_{[\mathrm{n}]} \right] \\
0 = \left(\mathbf{g}_{[\mathrm{n}]}^{\mathrm{T}} + \bar{\mathbf{x}}_{[\mathrm{n}]}^{\mathrm{T}} \mathbf{G}_{[\mathrm{n}]}^{\mathrm{T}} \right) + \bar{\mathbf{u}}_{[\mathrm{n}]}^{T} \mathbf{H}_{[\mathrm{n}]}^{\mathrm{T}} \\
\bar{\mathbf{u}}_{[\mathrm{n}]}^{T} \mathbf{H}_{[\mathrm{n}]}^{\mathrm{T}} = - \left(\mathbf{g}_{[\mathrm{n}]}^{\mathrm{T}} + \bar{\mathbf{x}}_{[\mathrm{n}]}^{\mathrm{T}} \mathbf{G}_{[\mathrm{n}]}^{\mathrm{T}} \right) \\
\mathbf{H}_{[\mathrm{n}]} \bar{\mathbf{u}}_{[\mathrm{n}]} = - \left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]} \right) \\
\bar{\mathbf{u}}_{[\mathrm{n}]} = -\mathbf{H}_{[\mathrm{n}]}^{-1} \left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]} \right) \\
= -\mathbf{H}_{[\mathrm{n}]}^{-1} \mathbf{g}_{[\mathrm{n}]} - \mathbf{H}_{[\mathrm{n}]}^{-1} \mathbf{G}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]} \\
= \mathbf{u}_{[\mathrm{n}]}^{\text{ff-inc}} + \mathbf{K}_{[\mathrm{n}]}^{\text{fb-inc}} \bar{\mathbf{x}}_{[\mathrm{n}]}
\end{aligned}
$$

$$
\mathbf{u}_{[\mathrm{n}]}^{\text{ff-inc}} = -\mathbf{H}_{[\mathrm{n}]}^{-1} \mathbf{g}_{[\mathrm{n}]} \qquad \texttt{feedforward increment:} \quad (N_u \times 1) \ \texttt{vector}
$$

$$
\mathbf{K}_{[\mathrm{n}]}^{\text{fb-inc}} = -\mathbf{H}_{[\mathrm{n}]}^{-1} \mathbf{G}_{[\mathrm{n}]} \qquad \texttt{feedback increment:} \quad (N_u \times N_x) \ \texttt{matrix}
$$

The optimal update policy is now

$$
\mathbf{u}_{[\mathrm{n}]new}^{*} = \mathbf{u}_{[\mathrm{n}]old}^{*} + \mathbf{u}_{[\mathrm{n}]}^{\text{ff-inc}} + \mathbf{K}_{[\mathrm{n}]}^{\text{fb-inc}} \bar{\mathbf{x}}_{[\mathrm{n}]}
$$

Substituting Running Cost $L_{[n]}$ and Value Function at $n+1$

$$
\begin{aligned}
V^*_{[n]}(\bar{\mathbf{x}}_{[n]}) = \min_{\mathbf{u}_{[n]}} & L^* + s_{[n+1]} \\
& + \left(\mathbf{L}^*_x + \mathbf{s}_{[n+1]}\mathbf{A}_{[n]}\right)\bar{\mathbf{x}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]}\left(\mathbf{L}^*_{xx} + \mathbf{A}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{A}_{[n]}\right)\bar{\mathbf{x}}_{[n]} \\
& + \left(\mathbf{g}^{\mathrm{T}}_{[n]} + \bar{\mathbf{x}}^{\mathrm{T}}_{[n]}\mathbf{G}^{\mathrm{T}}_{[n]}\right)\left(\mathbf{u}^{\text{ff-inc}}_{[n]} + \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}\right) \\
& + \frac{1}{2}(\mathbf{u}^{\text{ff-inc}}_{[n]} + \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]})^T\mathbf{H}^{\mathrm{T}}_{[n]}(\mathbf{u}^{\text{ff-inc}}_{[n]} + \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}) \\
= \min_{\mathbf{u}_{[n]}} & L^* + s_{[n+1]} \\
& + \left(\mathbf{L}^*_x + \mathbf{s}_{[n+1]}\mathbf{A}_{[n]}\right)\bar{\mathbf{x}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]}\left(\mathbf{L}^*_{xx} + \mathbf{A}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{A}_{[n]}\right)\bar{\mathbf{x}}_{[n]} \\
& + \left(\mathbf{g}^{\mathrm{T}}_{[n]} + \bar{\mathbf{x}}^{\mathrm{T}}_{[n]}\mathbf{G}^{\mathrm{T}}_{[n]}\right)\left(\mathbf{u}^{\text{ff-inc}}_{[n]} + \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}\right) \\
& + \frac{1}{2}\left((\mathbf{u}^{\text{ff-inc}}_{[n]})^{\mathbf{T}} + \bar{\mathbf{x}}^{\mathbf{T}}_{[n]}(\mathbf{K}^{\text{fb-inc}}_{[n]})^{\mathrm{T}}\right)\mathbf{H}^{\mathrm{T}}_{[n]}(\mathbf{u}^{\text{ff-inc}}_{[n]} + \mathbf{K}^{\text{fb-inc}}_{[n]}\bar{\mathbf{x}}_{[n]}) \\
= \min_{\mathbf{u}_{[n]}} & L^* + s_{[n+1]} + \left(\mathbf{g}^{\mathrm{T}}_{[n]} + \frac{1}{2}(\mathbf{u}^{\text{ff-inc}}_{[n]})^{\mathbf{T}}\mathbf{H}^{\mathrm{T}}_{[n]}\right)\mathbf{u}^{\text{ff-inc}}_{[n]} \\
& + \left(\mathbf{L}^*_x + \mathbf{s}_{[n+1]}\mathbf{A}_{[n]} + (\mathbf{u}^{\text{ff-inc}}_{[n]})^{\mathbf{T}}\mathbf{G}_{[n]} + \left(\mathbf{g}^{\mathrm{T}}_{[n]} + (\mathbf{u}^{\text{ff-inc}}_{[n]})^{\mathbf{T}}\mathbf{H}^{\mathrm{T}}_{[n]}\right)\mathbf{K}^{\text{fb-inc}}_{[n]}\right)\bar{\mathbf{x}}_{[n]} \\
& + \frac{1}{2}\bar{\mathbf{x}}^T_{[n]}\left(\mathbf{L}^*_{xx} + \mathbf{A}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{A}_{[n]} + (\mathbf{K}^{\text{fb-inc}}_{[n]})^{\mathrm{T}}\mathbf{G}_{[n]} + \left((\mathbf{K}^{\text{fb-inc}}_{[n]})^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}_{[n]} + \mathbf{G}^{\mathrm{T}}_{[n]}\right)\mathbf{K}^{\text{fb-inc}}_{[n]}\right)\bar{\mathbf{x}}_{[n]}
\end{aligned}
$$

Evaluate $\mathbf{S}$, $\mathbf{s}$, and s through backwards Riccati functions

$$
\begin{aligned}
\mathbf{S}_{[n]} &= \mathbf{L}^*_{xx} + \mathbf{A}^{\mathrm{T}}_{[n]}\mathbf{S}_{[n+1]}\mathbf{A}_{[n]} + (\mathbf{K}^{\text{fb-inc}}_{[n]})^{\mathrm{T}}\mathbf{G}_{[n]} + \left((\mathbf{K}^{\text{fb-inc}}_{[n]})^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}_{[n]} + \mathbf{G}^{\mathrm{T}}_{[n]}\right)\mathbf{K}^{\text{fb-inc}}_{[n]} \\
\mathbf{s}_{[n]} &= \mathbf{L}^*_x + \mathbf{s}_{[n+1]}\mathbf{A}_{[n]} + (\mathbf{u}^{\text{ff-inc}}_{[n]})^{\mathbf{T}}\mathbf{G}_{[n]} + \left(\mathbf{g}^{\mathrm{T}}_{[n]} + (\mathbf{u}^{\text{ff-inc}}_{[n]})^{\mathbf{T}}\mathbf{H}^{\mathrm{T}}_{[n]}\right)\mathbf{K}^{\text{fb-inc}}_{[n]} \\
s_{[n]} &= L^* + s_{[n+1]} + \left(\mathbf{g}^{\mathrm{T}}_{[n]} + \frac{1}{2}(\mathbf{u}^{\text{ff-inc}}_{[n]})^{\mathbf{T}}\mathbf{H}^{\mathrm{T}}_{[n]}\right)\mathbf{u}^{\text{ff-inc}}_{[n]}
\end{aligned}
$$

Initialize with terminal conditions

$$\mathbf{S}_{[\mathrm{N}]} = \mathbf{\Phi}^*_{xx} \qquad (N_x \times N_x) \quad \texttt{matrix}$$

$$\mathbf{s}_{[\mathrm{N}]} = \mathbf{\Phi}^*_{x} \qquad (1 \times N_x) \quad \texttt{vector}$$

$$s_{[\mathrm{N}]} = \Phi^* \qquad (1 \times 1) \quad \texttt{scalar}$$

A.1 SLQ with Inequality Constraints

When there is an inequality constraint, we have two possible strategies in choosing a control input: applying the optimal unconstrained control (works exactly as above, except that we have to check if it's valid), or applying an optimal control that takes the system exactly to the boundary, which we consider now. In the forward rollout, we will simply apply the control strategy that leads to the best allowed final state. (Note that there will always be at least one allowed final state, namely the one computed to end on the boundary.)

Now, assume we have some constraint function $\mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) \leq 0$. Define

$$\mathbf{C}_x = \left.\frac{\partial \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{x}}\right|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} \qquad (1 \times N_x) \quad \texttt{vector}$$

$$\mathbf{C}_u = \left.\frac{\partial \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\partial \mathbf{u}}\right|_{\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]}} \qquad (1 \times N_u) \quad \texttt{vector}$$

The necessary condition for $\bar{\mathbf{u}}_{[\mathrm{n}]}$ to be a constrained local minimum of the value function is:

$$\begin{aligned} 0 &= \left(\mathbf{g}_{[\mathrm{n}]}^{\mathrm{T}} + \bar{\mathbf{x}}_{[\mathrm{n}]}^{\mathrm{T}} \mathbf{G}_{[\mathrm{n}]}^{\mathrm{T}}\right) + \bar{\mathbf{u}}_{[\mathrm{n}]}^{T} \mathbf{H}_{[\mathrm{n}]}^{\mathrm{T}} + \lambda \mathbf{C}_u^{\mathrm{T}} \\ \mathbf{H}_{[\mathrm{n}]} \bar{\mathbf{u}}_{[\mathrm{n}]} &= -\left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]} + \lambda \mathbf{C}_u^{\mathrm{T}}\right) \\ \bar{\mathbf{u}}_{[\mathrm{n}]} &= -\mathbf{H}_{[\mathrm{n}]}^{-1}\left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]} \bar{\mathbf{x}}_{[\mathrm{n}]} + \lambda \mathbf{C}_u^{\mathrm{T}}\right) \end{aligned}$$

Linearly approximating the constraint, the condition of ending on the boundary tells us:

$$\begin{aligned} 0 &= \mathbf{c}(\mathbf{x}_{[\mathrm{n}]} + \bar{\mathbf{x}}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]} + \bar{\mathbf{u}}_{[\mathrm{n}]}) \\ 0 &= \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) + \mathbf{C}_x \bar{\mathbf{x}}_{[\mathrm{n}]} + \mathbf{C}_u \bar{\mathbf{u}}_{[\mathrm{n}]} \\ \mathbf{C}_u \bar{\mathbf{u}}_{[\mathrm{n}]} &= -\mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) - \mathbf{C}_x \bar{\mathbf{x}}_{[\mathrm{n}]} \end{aligned}$$

Since $\mathbf{C}_u$ isn't invertible (or even square) in general, we'll just left-multiply the local minimum condition by $\mathbf{C}_u$, allowing us to solve for λ:

$$\begin{aligned}
\bar{\mathbf{u}}_{[\mathrm{n}]} &= -\mathbf{H}_{[\mathrm{n}]}^{-1}\left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]}\bar{\mathbf{x}}_{[\mathrm{n}]} + \lambda\mathbf{C}_u^{\mathrm{T}}\right) \\
\mathbf{C}_u\bar{\mathbf{u}}_{[\mathrm{n}]} &= -\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]}\bar{\mathbf{x}}_{[\mathrm{n}]} + \lambda\mathbf{C}_u^{\mathrm{T}}\right) \\
-\mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) - \mathbf{C}_x\bar{\mathbf{x}}_{[\mathrm{n}]} &= -\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]}\bar{\mathbf{x}}_{[\mathrm{n}]} + \lambda\mathbf{C}_u^{\mathrm{T}}\right) \\
\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}\lambda &= -\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]}\bar{\mathbf{x}}_{[\mathrm{n}]}\right) + \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) + \mathbf{C}_x\bar{\mathbf{x}}_{[\mathrm{n}]} \\
\lambda &= \frac{1}{\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}}\left(-\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\left(\mathbf{g}_{[\mathrm{n}]} + \mathbf{G}_{[\mathrm{n}]}\bar{\mathbf{x}}_{[\mathrm{n}]}\right) + \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}) + \mathbf{C}_x\bar{\mathbf{x}}_{[\mathrm{n}]}\right) \\
&= \frac{-\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{G}_{[\mathrm{n}]} + \mathbf{C}_x}{\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}}\bar{\mathbf{x}}_{[\mathrm{n}]} + \frac{-\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{g}_{[\mathrm{n}]} + \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]})}{\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}}
\end{aligned}$$

Now, plugging this in to our expression for $\bar{\mathbf{u}}_{[\mathrm{n}]}$ and splitting out the terms containing $\bar{\mathbf{x}}_{[\mathrm{n}]}$, we can extract the feedforward control and the feedback matrix:

$$\begin{aligned}
\bar{\mathbf{u}}_{[\mathrm{n}]} &= -\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{G}_{[\mathrm{n}]}\bar{\mathbf{x}}_{[\mathrm{n}]} - \mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{g}_{[\mathrm{n}]} - \mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}\lambda \\
&= \left(-\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{g}_{[\mathrm{n}]} + \frac{-\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}(-\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{g}_{[\mathrm{n}]} + \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}))}{\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}}\right) \\
&+ \left(-\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{G}_{[\mathrm{n}]} + \frac{-\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}(-\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{G}_{[\mathrm{n}]} + \mathbf{C}_x)}{\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}}\right)\bar{\mathbf{x}}_{[\mathrm{n}]}
\end{aligned}$$

This gives constrained feedforward and feedback increments of:

$$\mathbf{u}_{[\mathrm{n}]}^{\text{ff-inc}} = -\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{g}_{[\mathrm{n}]} + \frac{-\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}(-\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{g}_{[\mathrm{n}]} + \mathbf{c}(\mathbf{x}_{[\mathrm{n}]}, \mathbf{u}_{[\mathrm{n}]}))}{\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}} \qquad \texttt{ff increment:} \quad (N_u \times 1) \quad \texttt{vector}$$

$$\mathbf{K}_{[\mathrm{n}]}^{\text{fb-inc}} = -\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{G}_{[\mathrm{n}]} + \frac{-\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}(-\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{G}_{[\mathrm{n}]} + \mathbf{C}_x)}{\mathbf{C}_u\mathbf{H}_{[\mathrm{n}]}^{-1}\mathbf{C}_u^{\mathrm{T}}} \qquad \texttt{fb increment:} \quad (N_u \times N_x) \quad \texttt{matrix}$$

The optimal update policy is now

$$\mathbf{u}_{[\mathrm{n}]new}^{*} = \mathbf{u}_{[\mathrm{n}]old}^{*} + \mathbf{u}_{[\mathrm{n}]}^{\text{ff-inc}} + \mathbf{K}_{[\mathrm{n}]}^{\text{fb-inc}}\bar{\mathbf{x}}_{[\mathrm{n}]}$$

A.2 Doing Simultaneous Methods Differently

This idea of SLQ planning can be applied to a legged robot, a system with inherently hybrid dynamics. The nominal gait of a legged system can be naturally modeled as a hybrid-dynamical system wherein a sequence of continuous swing phases are punctuated by discrete impact events. DDP and SLQ can be used to conform a nominal gait to the environment, adjusting both continuous control torques and discrete impulses alike. While traditional SLQ planners fight the battle of coarse vs. fine discretization, this work takes a novel approach at coarse trajectory discretization. Instead of linearizing along the entire step trajectory, the entire hybrid step is linearized into a single discrete action. While this approach may seem counter-intuitive to achieving stable controllers, the underlying method achieves highly dynamic movements for a system with narrowly-stable margins. By optimizing over control parameters that affect step location, motion, and control simultaneously, we turn SLQ into a tool that selects the "best achievable" sequence of foothold positions to traverse terrain in the context of how easy it is to control the system online (see Fig. 4.7).

A.3 Differences from other DDP Planners

This idea of SLQ planning can be applied to a legged robot, a system with inherently hybrid dynamics. The nominal gait of a legged system is can be naturally modeled as a hybrid-dynamical system wherein a sequence of continuous swing phases are punctuated by discrete impact events. SLQ can be used to conform a nominal gait to the environment, adjusting both continuous control torques and discrete impulses alike. While our work has elements similar to many existing works, it has the following distinctions from other DDP-variant planners:

(i) We solve for the gradient of *time of contact* t_c with respect to state $\frac{\partial t_c}{\partial \mathbf{x}}$ and control $\frac{\partial t_c}{\partial \mathbf{u}}$ to be used in determining the optimal controller. The time of contact is herein defined as the transition between the continuous phase ending and the discrete phase beginning, represented in this work as the moment in which the foot of a walking system

comes in contact with the ground. A fundamental problem with hybrid dynamics is that every small control adjustment comes with a change to t_c, which can be solved for as an additional optimization parameter [41]. We use a method that does not require solving for t_c for each incremental control adjustment, but instead solves for the the gradient of the hybrid step directly. We embed these partial derivatives into our SLQ formulation such that our control solution automatically considers Δt_c for every control adjustment $\Delta \mathbf{u}$. This eliminates the need to solve for additional parameters such as time of contact.

(ii) Rather than discretizing a continuous nominal trajectory into S locally linear segments followed by a discrete event, we discretize the entire hybrid step into one single movement. We do this by embedding a local continuous controller whose parameters $\mathbf{u}$ (gains, etc.) are tuned by the SLQ planner. This approach enables a simple PD controller to govern highly dynamic walking motions. For a finite N-step horizon, this method of linearization transforms a backwards Ricatti control updates from NS segments into N iterations, faster by a factor of S, shown in Fig. **??**. Our method becomes an N-step foothold optimization against the terrain cost map, combining both foothold selection and motion planning into a unified approach. In their work, [94] similarly tuned continuous control parameters (spring constant, etc.), but used an LQR controller for infinite time horizon gait transition and not an SLQ controller for finite time horizon footstep planning as we did. [95] did similar tuning using a deadbeat control approach for footstep placement of a spring-loaded inverted pendulum (SLIP) model, but this system allows "instantaneous joint angle placement of massless legs" and lacks the complexity of a variable inertia during the continuous spring phase that we consider with the Compass Gait Walker.

(iii) Our method solves for the gradients of the hybrid step using a numerical integration of the exact analytical gradient's time derivative using an ODE solver. For all practical purposes, this method yields the numerical equivalent to the exact solution for a system where the analytical solution is unknown, and the numerical approximation error should be at or better than that of any numerical gradient method (finite difference, automatic differentiation, etc.). Not only is this method more accurate, but it is faster

than conventional differentiation methods, as shown in Chap. 7.

Appendix B

Projected Hybrid Dynamic Gradient

In this appendix, the control gradient (4.11) for the projected hybrid dynamics referenced in (4.7) is derived. Although this derivation is specific to (4.7), the same method can be used generically to generate gradients for more complex hybrid systems. To keep the derivation compact, $|_\iota$ is used to represent the equation evaluation point $|_{t=t^-_{c[n+1]}}$. Parallel derivation can be done with respect to state gradient, whose results are shown in Section B.3.

B.1 Projected linearization with respect to changes in $\mathbf{u}_{[n]}$

We begin by applying the chain rule to the gradient of (4.9)

$$\begin{aligned}\frac{\partial P\left(\mathbf{x}_{[n]}, \mathbf{u}_{[n]}\right)}{\partial \mathbf{u}_{[n]}} &= \frac{\partial}{\partial \mathbf{u}_{[n]}} D\left(\mathbf{q}\left(t^-_{c[n+1]}, \mathbf{x}_{[n]}, \mathbf{u}_{[n]}\right)\right) \\ &= \frac{\partial D}{\partial \mathbf{q}}\left[f\left(\mathbf{q}(t)\right)\frac{\partial t^-_{c[n+1]}}{\partial \mathbf{u}_{[n]}} + \frac{\partial \mathbf{q}(t)}{\partial \mathbf{u}_{[n]}}\right]\Bigg|_\iota . \end{aligned} \tag{B.1}$$

Expanding $\left.\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}}\right|_{\iota}$ from (B.1)

$$\begin{aligned}\left.\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}}\right|_{\iota} =& \frac{\partial}{\partial \mathbf{u}_{[\mathrm{n}]}}\left[M\left(\mathbf{q}\left(\mathrm{t}_{\mathrm{c}[\mathrm{n}]}^{-}, \mathbf{x}_{[\mathrm{n}]}\right), \mathbf{u}_{[\mathrm{n}]}\right)+\int_{t_{c[\mathrm{n}]}^{+}}^{t} f\left(\mathbf{q}(\tau), \mathbf{u}_{[\mathrm{n}]}\right) d\tau\right]\Bigg|_{\iota} \\ =& \frac{\partial M}{\partial \mathbf{u}_{[\mathrm{n}]}}+\left[\frac{\partial}{\partial \mathbf{u}_{[\mathrm{n}]}} \int_{t_{c[\mathrm{n}]}^{+}}^{t} f\left(\mathbf{q}(\tau), \mathbf{u}_{[\mathrm{n}]}\right) d\tau\right]\Bigg|_{\iota}. \end{aligned} \tag{B.2}$$

When applying the Leibniz integral rule to the integral in (B.2)

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{u}_{[\mathrm{n}]}} \int_{t_{c[\mathrm{n}]}^{+}}^{t} f\left(\mathbf{q}(\tau), \mathbf{u}_{[\mathrm{n}]}\right) d\tau\Bigg|_{\iota} \\ &= \Bigg[f\left(\mathbf{q}(t), \mathbf{u}_{[\mathrm{n}]}\right) \frac{\partial t}{\partial \mathbf{u}_{[\mathrm{n}]}}-f\left(\mathbf{q}(t_{\mathrm{c}[\mathrm{n}]}^{+}), \mathbf{u}_{[\mathrm{n}]}\right) \frac{\partial t_{c[n]}^{+}}{\partial \mathbf{u}_{[\mathrm{n}]}} \\ &\quad+\int_{t_{c[\mathrm{n}]}^{+}}^{t}\left[\frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{q}}\left(\frac{\partial \mathbf{q}(\tau)}{\partial \tau} \frac{\partial \tau}{\partial \mathbf{u}_{[\mathrm{n}]}}+\frac{\partial \mathbf{q}(\tau)}{\partial \mathbf{u}_{[\mathrm{n}]}}+\frac{\partial \mathbf{q}(\tau)}{\partial \mathbf{x}_{[\mathrm{n}]}} \frac{\partial \mathbf{x}_{[\mathrm{n}]}}{\partial \mathbf{u}_{[\mathrm{n}]}}\right)+\frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{u}_{[\mathrm{n}]}} d\tau\right]\Bigg]\Bigg|_{\iota} \\ &= \left[0-0+\int_{t_{c[\mathrm{n}]}^{+}}^{t}\left[\frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{q}}\left(0+\frac{\partial \mathbf{q}(\tau)}{\partial \mathbf{u}_{[\mathrm{n}]}}+0\right)+\frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{u}_{[\mathrm{n}]}} d\tau\right]\right]\Bigg|_{\iota} \\ &= \left[\int_{t_{c[\mathrm{n}]}^{+}}^{t} \frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{q}} \frac{\partial \mathbf{q}(\tau)}{\partial \mathbf{u}_{[\mathrm{n}]}}+\frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{u}_{[\mathrm{n}]}} d\tau\right]\Bigg|_{\iota}, \end{aligned} \tag{B.3}$$

in which most of the expanded derivatives go to zero as seen in (B.3). Substituting (B.3) into (B.2) yields

$$\left.\frac{\partial \mathbf{q}(t)}{\partial \mathbf{u}_{[\mathrm{n}]}}\right|_{\iota}=\left[\frac{\partial M}{\partial \mathbf{u}_{[\mathrm{n}]}}+\int_{t_{c[\mathrm{n}]}^{+}}^{t} \frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{q}} \frac{\partial \mathbf{q}(\tau)}{\partial \mathbf{u}_{[\mathrm{n}]}} d\tau+\frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{u}_{[\mathrm{n}]}}\right]\Bigg|_{\iota}. \tag{B.4}$$

The second fundamental theorem of Calculus states

$$\begin{aligned}\frac{d}{dt} \frac{\partial \mathbf{q}(t)}{\partial \mathbf{u}_{[\mathrm{n}]}} &= \frac{d}{dt} \int_{t_{c[\mathrm{n}]}^{+}}^{t} \frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{q}} \frac{\partial \mathbf{q}(\tau)}{\partial \mathbf{u}_{[\mathrm{n}]}}+\frac{\partial f\left(\mathbf{q}(\tau)\right)}{\partial \mathbf{u}_{[\mathrm{n}]}} d\tau \\ &= \frac{\partial f\left(\mathbf{q}(\mathrm{t}), \mathbf{u}_{[\mathrm{n}]}\right)}{\partial \mathbf{q}} \frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}}+\frac{\partial f\left(\mathbf{q}(\mathrm{t}), \mathbf{u}_{[\mathrm{n}]}\right)}{\partial \mathbf{u}_{[\mathrm{n}]}}. \end{aligned} \tag{B.5}$$

Forward integration of the form $\dot{y}=Ay$ with initial conditions $\frac{\partial \mathbf{q}(t_{c[\mathrm{n}]}^{+})}{\partial \mathbf{u}_{[\mathrm{n}]}}=\frac{\partial M}{\partial \mathbf{u}_{[\mathrm{n}]}}$ of the form $y(0)=c_0$.

B.2 Solve for collision gradients

The goal in this section is to find the unknown term $\frac{\partial t^-_{c[n+1]}}{\partial \mathbf{u}_{[n]}}$ from equation (B.1). Set function ϕ to represent the end of integration

$$\phi\left(q\left(t^-_{c[n+1]}, \mathbf{x}_{[n]}, \mathbf{u}_{[n]}\right)\right) = 0. \tag{B.6}$$

Taking gradient of ϕ with respect to $\mathbf{u}_{[n]}$,

$$\frac{\partial}{\partial \mathbf{u}_{[n]}}\phi\left(q\left(t^-_{c[n+1]}, \mathbf{x}_{[n]}, \mathbf{u}_{[n]}\right)\right) = 0$$

$$\begin{aligned} 0 &= \left[\frac{\partial \phi}{\partial \mathbf{q}}\left(\frac{\partial \mathbf{q}(t)}{\partial \mathbf{u}_{[n]}} + \frac{\partial \mathbf{q}(t)}{\partial \mathbf{x}_{[n]}}\frac{\partial \mathbf{x}_{[n]}}{\partial \mathbf{u}_{[n]}} + \frac{\partial \mathbf{q}(t)}{\partial t}\frac{\partial t^-_{c[n+1]}}{\partial \mathbf{u}_{[n]}}\right)\right]\Bigg|_\iota \\ &= \left[\frac{\partial \phi}{\partial \mathbf{q}} f(\mathbf{q}(t))\frac{\partial t^-_{c[n+1]}}{\partial \mathbf{u}_{[n]}} + \frac{\partial \phi}{\partial \mathbf{q}}\frac{\partial \mathbf{q}(t)}{\partial \mathbf{u}_{[n]}}\right]\Bigg|_\iota \end{aligned} \tag{B.7}$$

and rearranging (B.7) reveals (B.8)

$$\frac{\partial t^-_{c[n+1]}}{\partial \mathbf{u}_{[n]}} = -\left[\left(\frac{\partial \phi}{\partial \mathbf{q}} f(\mathbf{q}(t))\right)^{-1}\left(\frac{\partial \phi}{\partial \mathbf{q}}\frac{\partial \mathbf{q}(t)}{\partial \mathbf{u}_{[n]}}\right)\right]_\iota. \tag{B.8}$$

B.3 Linearization of the Nonlinear Projected Dynamics

This linearization method solves for the gradients of the projected dynamics using a numerical integration of the exact analytical gradient's time derivative using an ODE solver. By doing so, this method yields the numerical equivalent to the exact solution for a system where the analytical solution is unknown, yielding a faster and more accurate result than numerical gradient methods (finite difference, automatic differentiation, etc.), as shown empirically in Chapter 7.

The sequence to solve for $\frac{\partial P}{\partial \mathbf{x}_{[n]}}$ and $\frac{\partial P}{\partial \mathbf{u}_{[n]}}$ can be solved in parallel using a single forward

integration of the dynamics, linearized about the pair $(\mathbf{x}^*_{[\mathrm{n}]}, \mathbf{u}^*_{[\mathrm{n}]})$. We begin by integrating

$$\dot{\mathbf{q}} = f(\mathbf{q}(\mathrm{t})) \tag{B.9}$$

$$\frac{d}{dt}\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{x}_{[\mathrm{n}]}} = \frac{\partial f\left(\mathbf{q}(\mathrm{t})\right)}{\partial \mathbf{q}}\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{x}_{[\mathrm{n}]}} \tag{B.10}$$

$$\frac{d}{dt}\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}} = \frac{\partial f\left(\mathbf{q}(\mathrm{t}), \mathbf{u}_{[\mathrm{n}]}\right)}{\partial \mathbf{q}}\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}} + \frac{\partial f\left(\mathbf{q}(\mathrm{t})\right)}{\partial \mathbf{u}_{[\mathrm{n}]}} \tag{B.11}$$

from initial conditions $M\left(D^{-1}\left(\mathbf{x}_{[\mathrm{n}]}\right)\right)$, $\frac{\partial M}{\partial \mathbf{q}}\frac{\partial D^{-1}}{\partial \mathbf{x}_{[\mathrm{n}]}}$ and $\frac{\partial M}{\partial \mathbf{u}_{[\mathrm{n}]}}$, respectively, until the next time of impact $t^-_{c[\mathrm{n+1}]}$ to reveal $f(\mathbf{q}(\mathrm{t}))$, $\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{x}_{[\mathrm{n}]}}$, and $\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}}$, evaluated at $t^-_{c[\mathrm{n+1}]}$. Second, solve

$$\xi = -\left[\left(\frac{\partial \phi}{\partial \mathbf{q}} f(\mathbf{q}(\mathrm{t}))\right)^{-1}\frac{\partial \phi}{\partial \mathbf{q}}\right]_{\iota} \tag{B.12}$$

$$\frac{\partial t^-_{c[\mathrm{n+1}]}}{\partial \mathbf{x}_{[\mathrm{n}]}} = \xi\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{x}_{[\mathrm{n}]}}\bigg|_{\iota} \qquad\qquad \frac{\partial t^-_{c[\mathrm{n+1}]}}{\partial \mathbf{u}_{[\mathrm{n}]}} = \xi\frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}}\bigg|_{\iota}. \tag{B.13}$$

Finally, solve (B.14) and (B.15) for the gradients with respect to state and control

$$\frac{\partial P}{\partial \mathbf{x}_{[\mathrm{n}]}} = \frac{\partial D}{\partial \mathbf{q}}\frac{\partial M}{\partial \mathbf{q}}\left[f(q(t))\frac{\partial t^-_{c[\mathrm{n+1}]}}{\partial \mathbf{x}_{[\mathrm{n}]}} + \frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{x}_{[\mathrm{n}]}}\right]\bigg|_{\iota} \tag{B.14}$$

$$\frac{\partial P}{\partial \mathbf{u}_{[\mathrm{n}]}} = \frac{\partial D}{\partial \mathbf{q}}\left[\frac{\partial M}{\partial \mathbf{q}}\left[f(q(t))\frac{\partial t^-_{c[\mathrm{n+1}]}}{\partial \mathbf{u}_{[\mathrm{n}]}} + \frac{\partial \mathbf{q}(\mathrm{t})}{\partial \mathbf{u}_{[\mathrm{n}]}}\right] + \frac{\partial M}{\partial \mathbf{u}_{[\mathrm{n}]}}\right]\bigg|_{\iota}. \tag{B.15}$$

Eqns. (B.11),(B.13),(B.15) yield the control gradient. A parallel set of derivations can be completed to yield the state gradient, whose results are listed in (B.10),(B.13),(B.14) above.

Appendix C

Regression for *Curve of Capture* and *Curve of Equal Energy*

This section contains the regression data to reconstruct the *curve of capture* and *curve of equal energy*. The regression data below matches the results from Figures 5.3, 5.5, and 6.4. The input regression data is normalized by its mean μ and standard deviation σ as follows.

$$E_*^- = \frac{E^- - \mu_E}{\sigma_E} \qquad \Delta z_*^- = \frac{\Delta z^- - \mu_{\Delta z}}{\sigma_{\Delta z}}, \tag{C.1}$$

where E_*^- and Δz_* represent the normalized input data. The respective mean and standard deviation data are located in Tables C.1-C.3. The normalized data can be taken to their respective power and multiplied by the coefficients of regression from the tables.

$$\Delta x = \sum_{m=0}^{O} \sum_{n=0}^{O} c_{m,n} \left(E_*^-\right)^m (\Delta z_*)^n \tag{C.2}$$

Where O represents the order of the system as listed in Sections 5.1, 5.2 and 6.1. Using (C.2) and information from Tables C.1, C.2 and C.3, the Δx footstep positions for (5.8), (5.10) and (6.3) can be recreated very quickly. The regression code is available at this link.

Table C.1: Coefficients for *Curve of Capture* Polynomial for Nonlinear Inverted Pendulum

power coefficient n of the step height $(\Delta z_*)^n$	0	1	2	3	4	5	6
6	-0.0002						
5	0.0271	0.022					
4	-0.1503	-0.1926	-0.0432				
3	0.2642	0.5905	0.3149	0.0419			
2	-0.1146	-0.6912	-0.7842	-0.2731	-0.0284		
1	-0.008	0.1626	0.6898	0.5426	0.1466	0.0129	
0	1.0236	-0.0883	-0.1225	-0.2628	-0.1736	-0.0449	-0.0042

power coefficient m of the scaled pre-collision energy $(E_*^-)^m$

	E^-	Δz
μ	14.5588	0.0283
σ	4.0225	0.2898

Table C.2: Coefficients for *Curve of Equal Energy* Polynomial for Nonlinear Inverted Pendulum

power coefficient n of the step height $(\Delta z_*)^n$	0	1	2	3	4
4	0.0133				
3	-0.0935	0.0001			
2	0.2021	0.0401	-0.039		
1	-0.275	0.0563	-0.0198	0.0498	
0	0.4126	-0.0507	-0.0681	-0.0018	-0.0088

power coefficient m of the scaled pre-collision energy $(E_*^-)^m$

	E^-	Δz
μ	18.2337	-0.2576
σ	8.0842	0.1616

Table C.3: Coefficients for *Curve of Capture* Polynomial for Compass Gait Walker

power coefficient n of the step height $(\Delta z_*)^n$	0	1	2	3	4	5
5	-0.0017					
4	0.0039	0.0012				
3	0.0018	-0.0049	-0.0012			
2	-0.0049	-0.0070	-0.0081	-0.0028		
1	0.0220	0.0483	0.0098	-0.0156	-0.0063	
0	0.3510	-0.1224	-0.0377	0.0075	0.0069	0.0009

power coefficient m of the scaled pre-collision energy $(E_{*[n-1]})^m$

	$E_{[n-1]}$	Δz
μ	60.7998	-0.1511
σ	34.8656	0.1566