University of California
Santa Barbara

Analyses and Robustness Quantification of Underactuated Biped Robot Locomotion

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Nihar Suresh Talele

Committee in charge:

Professor Katie Byl, Chair
Professor Andrew Teel
Professor João Hespanha
Professor Linda Petzold

December 2020

The Dissertation of Nihar Suresh Talele is approved.

Professor Andrew Teel

Professor João Hespanha

Professor Linda Petzold

Professor Katie Byl, Committee Chair

December 2020

Analyses and Robustness Quantification of Underactuated Biped Robot Locomotion

by

Nihar Suresh Talele

To my family and friends without whom this would not have been possible

Acknowledgements

There are many people who have helped me on this journey to get here but first and foremost, I would like thank my advisor Professor Katie Byl whose guidance has been invaluable throughout my research and my time in the lab. I am grateful for the opportunity to do research and be a part of very exciting and friendly atmosphere at the UCSB robotics lab which would not have been possible without her. Her continued support for the last five years has been invaluable in progress that I have made in my research.

I would also like to thank Professor Andrew Teel, Professor Joao Hespanha and Professor Linda Petzold for being a part of my committee and giving me valuable feedback on my research. In addition, the control courses they taught and the research discussions provided me with valuable insights on some promising directions in which to pursue my research. I would also like to acknowledge various faculty members from the electrical engineering department for offering interesting courses that I was glad to attend.

I would like to thank all my lab members past and present: Pat Terry, Chelsea Lau, Guilia Piovan, Sebastian Sovero, Cenk Oguz Saglam, Guillaume Bellegarda, Sean Gillen, Roman Aguilera, Thomas Ibbetson, Asutay Ozmen, Tom Strizic, Brian Satzinger for all their help over the years and making the UCSB robotics lab a fun and an exciting research environment. I would also like to thank my friends and colleagues outside the lab for making this journey at UCSB memorable.

Finally, I would like to thank Chitra for being a pillar of support and helping me to finish this journey.

Curriculum Vitæ

Nihar Suresh Talele

Education

2020	Ph.D. in Electrical and Computer Engineering (Expected), University of California, Santa Barbara.
2014	M.S. in Electrical Engineering, University of Southern California.
2011	B.E. in Electronics and Telecommunication Engineering, University of Pune.

Publications

- **Nihar Talele**. & Katie Byl. Mesh-based Tools to Analyze Deep Reinforcement Learning Policies for Underactuated Biped Locomotion. *arXiv*.
- **Nihar Talele**. & Katie Byl. Mesh-based Methods for Quantifying and Improving Robustness of a Planar Biped Model to Random Push Disturbances. In Proc. *American Control Conference (ACC)*, 2019.
- **Nihar Talele**. & Katie Byl. Methods and Performance Analyses for Design and Feedback Control of Efficient and Robust Planar Biped Walking. In Proc. *American Control Conference (ACC)*, 2019.
- Guillaume Bellegarda, **Nihar Talele**. & Katie Byl. Exploring Nonintuitive Optima for Dynamic Locomotion. In Proc. *ICRA*, 2018.
- Sebastian Sovero, **Nihar Talele**, Collin Smith, Nicholas Cox, Tim Swift & Katie Byl. Initial Data and Theory for High Specific Power Ankle Exoskeleton Device. In *Proc. ISER*, 2016.

Abstract

Analyses and Robustness Quantification of Underactuated Biped Robot Locomotion

by

Nihar Suresh Talele

Humanoid locomotion control is challenging due to the presence of underactuated dynamics, with constraints at the ground-foot contact imposing dynamic limitations on feasible motions. At the same time, deliberate underactuation in bipeds can potentially provide more energy efficient locomotion, making trade-offs between efficiency and stability a particularly interesting problem in biped control. Two approaches have become prominent in recent times for the control of legged locomotion. Model-based trajectory optimization has shown impressive results, for example in its application within the DARPA Robotics Challenge. Also, with the advent of improved computational capabilities, the field of deep reinforcement learning (DRL) is now being successfully applied to generate control policies for complicated systems like humanoids.

In the first part of this dissertation we use trajectory optimization methods to generate trajectories for a 5 link planar biped walker and control them via partial feedback linearization based controller. We perform experiments to demonstrate the importance of (a) considering and quantifying not only energy efficiency but also robustness of gaits, and (b) optimization not only of nominal motion trajectories but also of robot design parameters and feedback control policies.

In the second part we apply meshing tools to improve and analyze the performance of a 5-link planar biped model to random push perturbations. Creating a mesh for a 14-dimensional state space would typically be infeasible. However, as we show in this dissertation, low level controllers can restrict the reachable space of the system

to a much lower dimensional manifold, which makes it possible to apply our tools to improve the performance. We demonstrate the effectiveness of our tools by performing simulations on both: trajectories generated via optimization and policies generated using deep reinforcement learning.

This thesis includes the following publications:

- Sovero S., Talele N., Smith C., Cox N., Swift T., Byl K. (2017) Initial Data and Theory for a High Specific-Power Ankle Exoskeleton Device. In: Kulić D., Nakamura Y., Khatib O., Venture G. (eds) 2016 International Symposium on Experimental Robotics. ISER 2016. Springer Proceedings in Advanced Robotics, vol 1. Springer, Cham.

- ©IEEE. Reprinted, with permission, from Nihar Talele and Katie Byl, Methods and Performance Analyses for Design and Feedback Control of Efficient and Robust Planar Biped Walking, *American Control Conference (ACC)*, July 2019.

- ©IEEE. Reprinted, with permission, from Nihar Talele and Katie Byl, Mesh-based Methods for Quantifying and Improving Robustness of a Planar Biped Model to Random Push Disturbances, *American Control Conference (ACC)*, July 2019.

- Nihar Talele and Katie Byl, Mesh-based Tools to Analyze Deep Reinforcement Learning Policies for Underactuated Biped Locomotion, *arXiv*.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Practical robot locomotion needs to be both energy efficient and robust to variability and uncertainty. Both energy efficiency and robustness have long been goals for robot walking, and both mechanical design and control strategy play important roles in each objective. Even though walking may appear to be a trivial task for human beings, humanoid locomotion control is challenging due to the presence of underactuated dynamics, with constraints at the ground-foot contact imposing dynamic limitations on feasible motions. At the same time, deliberate underactuation in bipeds can potentially provide more energy efficient locomotion, making trade-offs between efficiency and stability a particularly interesting problem in biped control. Flat footed walking is easier to achieve due to lack of underactuated motion, at the same time, the motion generated is very inefficient as is evident by their significantly higher cost of transport. On the other hand, motions that involve underactuation are harder to control in practice and they are much more sensitive to external disturbances.

When it comes to energy, wheeled locomotion is much more efficient as compared to biped locomotion. So why study biped locomotion? The most important advantage offered by legged locomotion is the versatility in terms of the terrain that can be traversed.

Wheels can't be used to traverse rough terrain or climb mountains or even stairs. Humans can traverse a wide variety of terrain where only intermittent footholds are possible. The society around us has been built with humans in mind and if we want robots to assist us and operate efficiently in this environment, it would be advantageous to equip them with the capability of legged locomotion. Robots capable of legged locomotion could also be used for safety critical and hazardous tasks such as rescue operations and space exploration which can be very risky for humans. Humanoid robots would also be more effective in assisting the elderly. Advances in legged locomotion also enable physically disabled people to regain their lost motion through the use of prosthetic limbs and exoskeletons. Because of these, legged locomotion continues to be an active and an exciting area of research in robotics. The two most popular humanoids currently are shown in Fig. 1.1 and 1.2. Asimo is a humanoid robot by Honda and Atlas is made by Boston Dynamics.

While most of the research in this thesis focuses on biped locomotion, the tools and techniques developed are applicable to a wide variety of dynamical systems. The study of underactuated systems is an interesting area of research as such systems possess less actuators than the number of degrees of freedom to be controlled making it a challenging problem. This thesis focuses on analyzing the trade-off between energy efficiency and robustness for such underactuated systems as well as the quantification of robustness for such systems with limit cycle behavior.

1.1 Literature Review

Toward improved mechanical design, biped robots built on passive dynamic principles drew significant attention over a decade ago [4], but their success at reducing required energy has seemed to be coupled with fragile dynamics, yielding susceptibility

Figure 1.1: Humanoid robot Asimo by Honda. Image taken from [1]

.

to falls. Design of mechanical properties, i.e., lengths and mass distribution, clearly play an important role in enabling efficient legged locomotion, but they also arguably affect stability.

To improve controlled walking strategies, a range of work has focused on both trajectory optimization and control theory. Trajectory optimization through direct colloca-

Figure 1.2: Atlas robot by Boston Dynamics. Image taken from [2]

.

tion [5] is one promising approach. In 1999, for example, Hardt et al. formulated the problem of minimizing energy of a planar 5-link biped, both with and without ankle torque, using DIRCOL software [6] to solve a nonlinear optimization subject to contact constraints. Two years later, Paul et al. looked at simultaneous optimization of both mass distributions (robot design) and nominal motion trajectories, to be tracked via a simple proportional controller (with saturation limits), using simple neural networks to learn efficient locomotion [7].

In 2002, Westervelt and Grizzle highlighted the importance of optimizing walking motions while simultaneously guaranteeing asymptotic stability [8], as opposed to a still-dominating paradigm of sequential design, first optimizing a nominal trajectory and

subsequently adding feedback control in a more ad hoc way. As in [6], they also use DIRCOL, and they solve a sequential quadratic programming (SQP) problem to optimize the sum of u^2 across all four actuators. Note that [8] uses a hybrid zero dynamic (HZD) approach, which parameterizes joint trajectories on a monotonic, geometric variable. In a similar spirit, [9] produce energy-optimal gaits for the 5-link walker using polynomial trajectories in which the gait is defined as $q(s)$, as a function of geometry rather than time, by solving for optimal polynomial coefficients.

Various works have instead focused on optimizing robustness. Dai and Tedrake [10] optimized a measure of robustness that quantifies variation from a nominal trajectory during rough terrain locomotion, for both the spring-loaded inverted pendulum (SLIP) and compass gait (CG) walker planar legged locomotion models. In [11] Kuindersma et al. use quadratic programming to incorporate constraints in the feedback policy. In [12], Majumdar and Tedrake use the concept of funnel libraries to increase the robustness of optimal trajectories. In [13], Nguyen, Grizzle, Sreenath et al. use 2-step gait optimization and gait interpolation to navigate across rough terrain. In [14], Byl and Tedrake use the concept of metastability to quantify the robustness of simple rough-terrain walking models, proposing mean time to falling, or mean first-passage time (MFPT), as an important metric of merit. In [15], Saglam and Byl build on the previous work by analyzing the trade-offs between energy and robustness for a 5-link biped model using once-per-step switching between low-level sliding mode controllers for walking on rough terrain. As the approach models a discretized approximation of the resulting dynamics as a Markov decision process (MDP), feasibility of the methods in [15] depends upon the "meshability" of the resulting system; i.e., the stochastic system dynamics must visit only a relatively low-dimensional manifold within the full (i.e., 10-D set of positions and velocities, in [15]) state space, to avoid the curse of dimensionality.

Recent work by Hamed, Buss and Grizzle also focuses on robustness, tuning control

parameters to ensure not only stable eigenvalues of the Jacobian of the period-one return map of limit-cycle walking but also reduced sensitivity of this Jacobian to parameter variation [16]. Here, they decouple the selection of a nominal periodic orbit from that of optimizing a parameterized controller, e.g., torques include both the necessary feedforward terms exactly compatible with the limit cycle of interest, along with some flavor of feedback law (e.g., perhaps but not necessarily HZD) that has no effect along the exact limit cycle trajectory.

Finally, a few other recent works emphasize applicability of legged locomotion optimization to an expanding range of problems. Recent work by Ma, Hereid, Hubicki and Ames on the DURUS robot employs the HZD framework to optimize energy efficiency for stable 3D walking [17]. Xi, Yesilevskiy and Remy employ direct collocation (DC) to optimize energetic cost for gaits without a prescribed sequence of foot contacts with the grounds [18], and within our own group, we have used trajectory optimization to predict the theoretical cost of added mass in exoskeleton design [19] and to discover nonintuitive locomotion strategies for an underactuated, acrobot-based rolling system [20].

With the advent of improved computational capabilities, the field of deep reinforcement learning (DRL) is now being successfully applied to generate control policies for complicated dynamical systems like humanoids [21]. Recent advances in the deep reinforcement learning algorithms [22], [23] have shown the potential of this tool in generating robust control policies for a wide variety of tasks for complex underactuated systems. Although successful, deep reinforcement learning methods that use a simple reward function generate motions very non-human like and highly sterotyped. In [24], Merel, Tassa et. al. use limited demonstrations consisting only of partially observed state features to generate more human like movement patterns. OpenAI Baselines [25] is a collection of various deep reinforcement learning algorithms like PPO [23] that have been used to generate robust control policies for humanoid locomotion in the presence of diturbances.

Some recent works like [26] combine both trajectory optimization as well as learning approaches to obtain an overall improvement in robustness of the control policies for locomotion. Within our own group we have combined trajectory optimization and deep reinforcement learning [27] to leverage the advantage of both approaches to obtain an overall better control policy.

1.2 Contributions and Goals

Although significant research has been done in the field of legged locomotion, there are still quite a few issues that need to be addressed before walking robots become commonplace in real life. The most primary among them is the quantification of robustness for a robotic system with biped locomotion capabilities. In [14] Byl and Tedrake try to address this issue using the concept of metastability. This is done by completely exploring the reachable state space of the system on a Poincaré section and building a Markov Decision Process.

In this thesis, we focus on several related, open challenges in simultaneously optimizing for energetics and robustness. We highlight important choices made in differentiation and integration that improve speed and accuracy, since local optimization provides only approximate results. With an aim toward improving both energetics and robustness, we explore how variations in mass distribution affect metrics for each of these goals and observe a natural trade-off (between metrics) that results from tuning of feedback control. Our results demonstrate that choice of both mass distribution and feedback control structure have important, and apparently coupled, effects on both energy use and stability, providing evidence for the hypothesis that more comprehensive frameworks are needed for simultaneous optimization across system parameters and desired metrics.

We also extend the applicability of meshing tools to analyze the effects not only of

"noise" in the terrain height, which determines when an otherwise-deterministic continuous-time trajectory ends (i.e., with swing leg impact at an unknown ground height), but also of more general perturbation disturbances typical of real-world scenarios, such as push disturbances that can happen at any time during the gait cycle, as opposed to only determining when an otherwise-deterministic continuous-time trajectory ends (i.e., with swing leg impact at an unknown ground height). Additionally, we demonstrate applicability of these tools for other low-level control schemes, aside from sliding mode control (SMC), to explore whether the finite-time convergence of SMC to lower-dimensional manifolds is a critical key to meshability; we conclude that it is not.

Finally, we also demonstrate the applicability of our tools to quantify the robustness of deep reinforcement learning control policies for biped systems. The field of deep reinforcement learning is expanding rapidly and is becoming increasingly popular as a tool to obtain control polices for dynamical systems. However, there are very few tools available to quantify the performance of these policies. We show how our meshing tools can be applied for such purposes by analyzing and quantifying the performance of DRL trained control policies for 5 link planar biped model in the presence of push disturbances.

1.3 Organization of Thesis

The next chapter focuses on the optimization tools on which most of the motion planning in this thesis is based and an explanation on the implementation of the optimal trajectories on system using a low level controller. Chapter 3 focuses on the simulation studies of added mass on human body and their effects on the cost of transport to validate the human studies that were performed to analyze the optimal location to place the exoskeleton on humans. In Chapter 4 we present results on the effect of system

design parameters and choice of feedback control policies on the energy consumption and robustness for biped locomotion. Chapter 5 focuses on the extension of our meshing tools to more real world noise scenarios such as push disturbances that can happen at anytime during the gait cycle. In Chapter 6 we see how our meshing tools can be used to quantify the performance of DRL control polices for biped locomotion. We conclude in Chapter 7 by outlining the tasks for future experiments.

Chapter 2

Trajectory Optimization

2.1 Control as an Optimization Problem

Control of underactuated systems like legged robots is not trivial. Traditional control techniques such as partial feedback linearization can generate motions for such systems, however, the kind of motions that can be generated are very restrictive. Because of which, motion planning for such systems can be split into two parts: Generating the reference signal and controlling the system to this reference signal. Biped robots are complicated non linear dynamical systems due to the complex interconnection of various links. This makes it harder to come up with a reference trajectory for such systems. To solve this problem, numerical methods such as trajectory optimization are applied. Trajectory optimization poses the problem of finding a control trajectory for a system as a long term optimization of a certain specific scalar cost function. This function can be any function of the system dynamics under consideration. The structure of this formulation makes it applicable to a wide variety of problems. Biped robots are systems with a large number of DOF and complicated non linear dynamics because of which it becomes difficult to obtain globally optimal solutions. For this reason, the optimization

techniques used in this thesis focus on locally optimal solutions.

Of the vast majority of solvers available for solving optimization problems, the two most popular solvers for non linear optimization are the interior point solvers and sequential quadratic programming(SQP) solvers. These methods are covered in detail in [28] and [29] so we will just provide a brief overview here. Sequential quadratic programming involves solving a series of QP problems sequentially by doing a second-order approximation for the objective function and a first order approximation for the constraints. Interior point methods, on the other hand, work by iteratively solving a series of unconstrained problems by making the constraints a part of the cost function with help of barrier functions. The name 'Interior point methods' stems from the fact that all the successive solutions of the optimization problem are always inside the feasible region. This may prove useful for systems with unstable modes to avoid those operating regions with the help of constraints. Both interior point and SQP methods have shown to work well on large scale nonlinear optimization problems in practice. There are some situations however where one may perform better than the other. SQP methods are more robust to scaling issues as compared to Interior point methods. On the other hand, SQP methods less efficient if the number of active constraints are significantly less than number of open variables. Interior methods show their strength in large scale applications where they often though not always outperform SQP methods.

In order to apply these numerical optimization techniques to generate motion for dynamical systems, it is necessary to discretize the system dynamics on time. This means that the entire trajectory is split into a series of knot points. This discretization determines the accuracy of the solution as well as the speed of computation. The choice of integration scheme used also plays an important factor in the accuracy of the solution. Higher order integration schemes will lead to more accurate results but might be to expensive from computation standpoint. Another important consideration is the use of

implicit vs explicit integration schemes. Implicit methods are inherently stable however difficult to implement. However, some formulations allow for easier implementations of implicit methods as we shall see in the next sections. From a practical standpoint, once we obtain the trajectory we then have to use a low-level controller in order to implement this trajectory on the system. Details about the low-level controller will be covered in the next chapter.

2.2 Polynomial Trajectories

The most straight forward way to formulate control as an optimization problem is to represent the trajectories as an n^{th} degree polynomial and have the coefficents of the polynomial as an open variables. A slight modification of this approach would be to represent the trajectories using polynomials but instead of leaving the coefficients as open variables, the parameters that determine the coefficients such as the start and end conditions can be left as the open variables. An example of such parameterization is shown in Fig. 2.1. Here, a 4^{th} degree polynomial is used to represent a trajectory for a specific joint on a robot. As this is a 4^{th} degree polynomial, there are 5 open variables but instead of coefficients the five open variables were the starting and ending positions q_0, q_2, the starting and ending velocities $\dot{q_0}, \dot{q_2}$ and the peak position q_1. This approach was used in our optimization for the human simulation data that will be discussed in detail in chapter 3. In practice we observed that this approach works better than the straight forward way of having coefficients as the open variables.

The simplicity of this approach makes it easy to implement without any hassle. There are however quite a few drawbacks of using this approach. Having a low degree of polynomial representing a trajectory reduces the number of open variables the optimizer needs to solve for which makes the optimization faster but at the same time it results

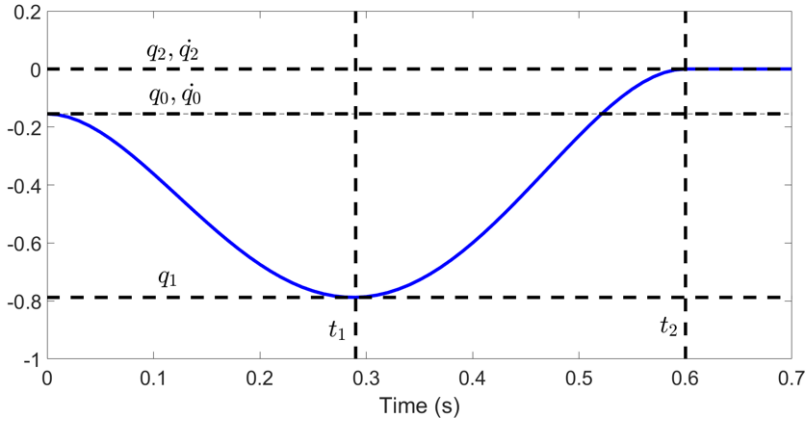

Figure 2.1: Parameterization of a polynomial used for representing a trajectory.

in a very restrictive solution. Higher-degree polynomials can be used to circumvent the issue but there are other more efficient methods of doing so as we will discuss in the next section.

2.3 Single Shooting Methods

In single shooting methods, we parameterize the inputs to the system using a first-order hold method and then use these values of the inputs at each knot point as the open variables for optimization. The optimization problem is then formulated as

$$\min_{u} \quad \sum_{n=0}^{N} g(x_n, u_n) \tag{2.1}$$

$$\text{such that} \quad \mathbf{\Phi}(x, u) \leq 0. \tag{2.2}$$

Here, the objective function is some scalar cost function of system dynamics such as energy and $\mathbf{\Phi}(x, u)$ is set of constraints under which we want the optimizer to find the solution. The constraints can be both linear as well as non linear. The important thing to note is that the open variables for optimization are the input values at each instant

of time. This formulation is very simple and allows the trajectories to be less restrictive without over complicating as would be the case with higher-degree polynomials. This method works well in practice for a large variety of systems. There is however a class of systems for which this method is not always successful. In order for the optimizer to calculate the gradient of the objective function and constraints w.r.t the open variables, it has to first evaluate the entire trajectory which is a function of the input u. If the system has any unstable modes, the system states might escape to infinity before the entire trajectory is rolled out. This in turn would make it infeasible to calculate the value of the gradient at the given point or iteration resulting in non convergence of the optimizer. Even if we impose constraints to prevent the system from going into unstable modes, the optimizer would need to calculate the gradient of those constraints for which it is necessary to evaluate the entire trajectory. To avoid these problems, we use multiple shooting and direct transcription methods.

2.4 Multiple Shooting Methods

In single shooting methods, the only open variables to the optimization are the inputs to the system. In direct transcription and multiple shooting methods, in addition to inputs we also have the states as open variables. This changes the problem formulation to

$$\min_{x,u} \quad \sum_{n=0}^{N} g(x_n, u_n) \tag{2.3}$$

$$\text{such that} \quad x_{n+1} = x_n + f(x_n, u_n)dt \quad \forall n \in [0, N] \tag{2.4}$$

$$\mathbf{\Phi}(x, u) \leq 0. \tag{2.5}$$

As we see, the objective function is now a function of both the state as well as the input to the system. In addition, we also need to apply an extra constraint as shown in equation 2.4. This is to ensure that all the states are physically consistent with the system dynamics. This increases the size of the optimization problem due to many more constraints and open variables.

Even though adding states as open variables increases the problem size significantly, this method works very well in practice. The advantage of having the states as open variables is that now we can upper bound and lower bound the states and thereby prevent the system from operating in any unstable modes. This is very important for systems which are inherently unstable such as legged robots. Another advantage provided by this formulation is that it results in matrices that are sparse. This makes the computation of the underlying linear system of equations very efficient. Single shooting methods on the other hand lead to matrices which are very dense. A third advantage provided by this formulation is that it makes it very convenient to used implicit methods of integration. Implicit methods calculate next state as a function of the current state and the next state. In multiple shooting and direct transcription methods, the entire state information is readily available making the implementation of implicit methods very convenient. Implicit methods are stable which is important for optimizer to converge at a solution. The constraint equation 2.4 is an example of a simple forward Euler integration which is an explicit method. In practice for our experiments in this thesis we have made use of the following integration scheme.

$$x_{k+1} = x_k + \frac{dt}{2}[f(x_k, u_k) + f(x_{k+1}, u_k)] \tag{2.6}$$

This is a zero order hold trapezoid integration scheme which is an implicit method

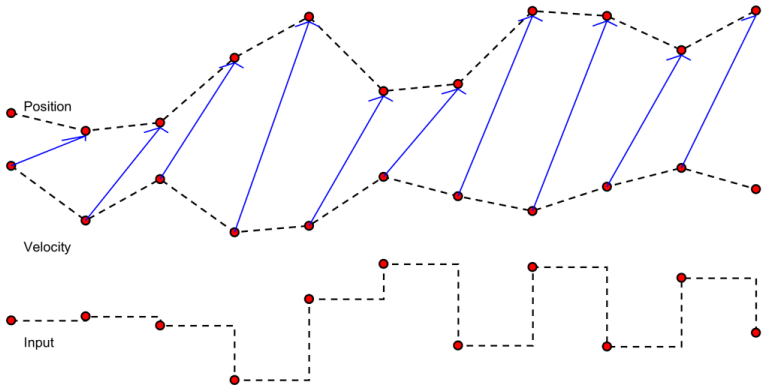

Figure 2.2: Parameterization of a traracectory for optimization using direct transcription method.

because we see that the next state is a function of the current state as well as the next state. Empirically, we observed that zero-order hold for input worked better than first-order hold as the latter resulted in trajectories which were very non smooth. It was possible to achieve smooth trajectories using first-order hold method but in practice the zero-order hold method still worked better.

An example of direct transcription parameterization for a trajectory is shown in Fig. 2.2, The red dots represent the values of position, velocity and input at the knot points. The blue arrows indicate the constraints applied to the system to make it feasible.

2.5 Cost of Transport

Trajectory optimization calculates the solution of the problem by iteratively reducing a certain scalar cost function. This cost function can be any function of the system dynamics. The most commonly used cost function for trajectory optimization is energy. Another popular cost function is the squared input to the system. A smooth cost function is necessary for the optimization to converge to a good solution. For most of our results

in this thesis, the cost function we have used is the cost of transport or COT. Cost of transport is defined by

$$COT = \frac{E}{Mgd} \tag{2.7}$$

Here E is the energy expended for the resulting motion, M is the mass of the system, g is the acceleration due to gravity and d is the distance travelled as a result of the locomotion. As we discretize the system dynamics on time in order to generate motion using trajectory optimization, we implement the COT using trapezoid rule as follows

$$COT = \frac{\sum_{k=0}^{N-1}\sum_{n=1}^{m}\widetilde{P}_n(k)\Delta t}{Mgd} \tag{2.8}$$

where there are N discrete time steps $t(k)$, d is the stride length M is the mass of the model and m is the total number of joints on the robot. Rate of work (power) at joint n during time step k is approximated as

$$\widetilde{P}_n(k) = \frac{\widetilde{P}_{n,1}(k) + \widetilde{P}_{n,2}(k)}{2} \qquad n = \{1, 2, ..., m\}, \tag{2.9}$$

where $\widetilde{P}_{n,1}(k)$ is the regularized version of $P_{n,1}(k) = \tau_n(k)\omega_n(k)$ and $\widetilde{P}_{n,2}(k)$ is the regularized version of $P_{n,2}(k) = \tau_n(k)\omega_n(k+1)$. Also, $\widetilde{P}_{n,i}(k)$ is defined to penalize both the positive as well as the negative mechanical work. We take the absolute value of the power at each joint however, this results in a non smooth cost function. To solve this problem, we instead smooth the cost function by using a regularization factor, so that

$$\widetilde{P}_{n,i} = \sqrt{P_{n,i}^2 + \epsilon^2} \qquad i = \{1, 2\} \tag{2.10}$$

as suggested in [30]. Unless otherwise mention, we use $\epsilon^2 = 0.01$, which works well for our optimization. Lower values of ϵ led to stability issues with the solver. Even though

the regularization introduces minor inaccuracies in the calculation of the energy, the smoothness obtained as a result is worth the tradeoff.

2.6 Control of Optimal Trajectories

The trajectories obtained from optimization are open loop trajectories. Ideally, these trajectories should work on the actual system to generate the desired motion. However, even with perfect knowledge of system dynamics, this would not be the case. This is due to the discretization introduced during the optimization process.

The most popular approach in recent times to implement the optimal trajectories on the system is by using LQR controller. In order to implement the LQR controller, we linearize the system dynamics at the curent operating point. In practice we found that, this type pf control approach works very well for systems that don't have impacts. But for our experiments on the 5 link planar biped model as described in Appendix A, this approach did not work well. Specifically, we linearized about each "knot point" of the optimal solution, and then controlled motions using the nominal feedforward torque (U_{ff}) added to feedback of the form $U_{fb} = -K(X - X_{nom})$, where $K = K(t)$ and $X_{nom} = X_{nom}(t)$ were interpolated between their values at the discrete points of the optimal solution. During continuous motion, the trajectories definitely converged toward the nominal trajectories as expected, but the effects through impacts were too destabilizing, resulting in falls after 3 to 6 steps.

In order to stabilize the optimal trajectories on the model, we use partial feedback linearization (PFL) based control scheme. The total input to the system is $U = U_{ff} + U_{fb}$ where

$$U_{fb} = (SD^{-1}B)^{-1}(v + SD^{-1}(C\dot{q} + G)), \tag{2.11}$$

and S is given by

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

and u_{ff} contains the feedforward torques compatible with the nominal dynamics. Our point foot walker is an underactuated model. We formulated our PFL control law using this structure in order to control all the degrees of freedom using only 4 actuators. The matrix S gives us this flexibility to control all the degrees of freedom in the way we want.

Given a passive contact of the stance leg with the ground, PFL allows us to directly set the accelerations of 4 out of 5 angles using v. We set $v = [\ddot{q}_{2des}, \ddot{q}_{1des}+\ddot{q}_{3des}, \ddot{q}_{4des}, \ddot{q}_{5des}]^{\mathsf{T}}$ where $\ddot{q}_{des} = -K_p(q - q_{des}) - K_d(\dot{q} - \dot{q}_{des})$. We set $K_p = \omega_n^2$ and $K_d = 2\zeta\omega_n$. For all our simulations we set $\zeta = 1$ and test across a range of ω_n values. We get $U_{ff},\ q_{des},\ \dot{q}_{des}$ by interpolating the trajectories from the optimization framework.

2.7 Gradients

An important consideration for these gradient based methods is the availability of good quality gradients. Non gradient based methods such as Nelder-Mead algorithms work relatively well for smaller problems. But as the dimensionality of the problem increases, gradient based methods perform much better as compared to search based methods. As mentioned though, the performance of gradient based methods is very sensitive to the quality of the derivatives. Most solvers these days are capable of numerically calculating the gradients using finite differencing methods. These methods are not that accurate however and can lead to inaccurate results or even in many cases fail to converge to a solution. To avoid this problem we use CasADi [31] which uses algorithmic

differentiation to calculate accurate gradients to machine precision. For our solver we use IPOPT [32], [33] which is an interior point based solver. The choice to use IPOPT was based on our own experimental results and the benchmarks from [34]. Based on our experiments we saw a significant improvement in the convergence rate for the solver when using algorithmic differentiation for gradients vs. numerical methods.

Chapter 3

High Specific-Power Ankle Exoskeleton Device

Exoskeletons have been a major thrust of robotics research since quite some time now. Even though significant progress has been made, conventional exoskeleton designs have been unable to provide metabolic benefit for day-to-day activities such as running and in many cases even walking as well. Recent work under DARPA's warrior web program [35] has made significant strides in this area. However, there are still a lot of challenges that need to be overcome before exoskeletons reach their full potential. One important stride in this direction came from the work at MIT [36] which has demonstrated metabolic benefit in walking. While this by itself is a very significant result, the most valuable output from this work is the notion of the augmentation factor equation. While this equation does not define the full system dynamics itself, it gives a metric to measure the performance of an exoskeleton. The equation allows us to quantify the benefit obtained by the added power of the skeleton while taking in to consideration the penalty accrued due to the added mass of the exoskeleton itself. In this chapter, we build off the structure of the augmentation factor by analyzing the effects of the mass of the exoskeleton and

the speed of the human gait on its two components: Added power and Power to carry. Specifically, we will be focusing on the simulation studies highlighting the effect of added mass on the human body. The main goal of this work is to corroborate the human experimental data with simulation as the former is always prone to large variances and inaccuracies and to provide more confidence in the validity of the hypothesis.

3.1 Augmentation Factor

The augmentation factor equation as shown below in Eq. 3.1 consists of two major components:

$$AF = \underbrace{\frac{p^+ + p_{dis}}{\eta^+}}_{AP} + \underbrace{\sum_i \beta_i m_i}_{PC}. \tag{3.1}$$

The first component, added power (AP) denotes the metabolic benefit provided by the exoskeleton. It takes into account the positive power p^+ added by the exoskeleton, the dissipated power p_{dis} and the muscle efficiency η^+. The second component of the augmentation factor is the power to carry(PC) which represents the burden incurred due the added mass of the exoskeleton. Here m_i is the mass of the exoskeleton added and i is the location at which it is added.

3.2 Simulation Model and optimization

For our simulation studies we used the following model as shown in Fig.3.1 This is a 7-link planar underactuated model with curved feet. The curved feet were added to simulate the rolling motion of the human feet while walking. The model has a total of 6 actuators: 2 at the hips, 2 at the knees and 2 at the ankles. The physical parameters for the model such as mass and inertia were taken from [37] such that the model resembles

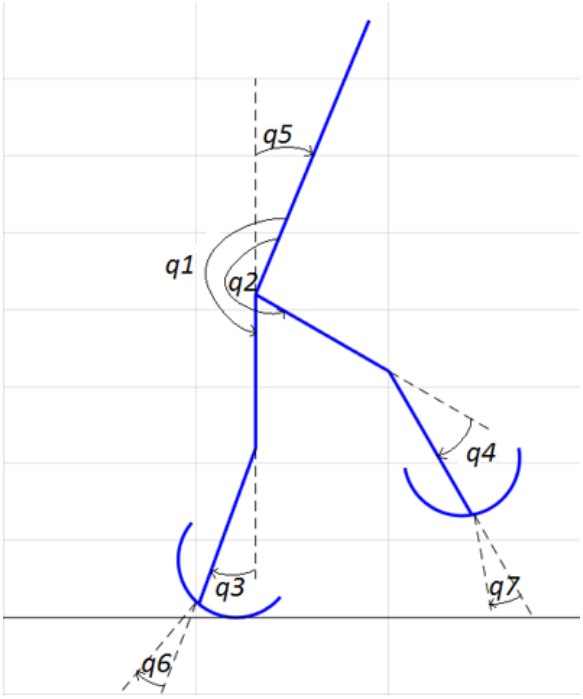

Figure 3.1: 7-link planar walker model.

a human being with a mass of 74.2 kg. The model was created in matlab and all our experiments were also performed in the same. Even though the model has curved feet, the model is still underactuatd as there is still a point contact between the feet and ground. The model has 7 degrees of freedom as shown where the angle of the torso q_5 is taken as an absolute angle with respect to the reference and all other angles are relative and measured w.r.t. the torso. We model the contact dynamics with the ground using an inelastic collision, i.e., part of the energy is lost when the swing foot impacts the ground.

For trajectory optimization, we parameterize the joint angle trajectories as polynomials shown in Fig. 2.1. We perform the optimization in matlab using using the interior point method in fmincon. We optimize for the cost of transport and the trajectories were generated with a constraint that the motion is a limit cycle i.e., the states go through the same evolution every step.

3.3 Added Mass Study

To understand the effect of added mass in the augmentation factor equation, we perform a series of experiments by placing additional masses on the model during walking and measuring the increase in the cost of transport as a result. We perform two sets of experiments to study this effect. In the first experiment we study the effect of adding mass on the human body and walking at a constant velocity. The red dots in the Fig. 3.2 shows the locations at which the additional mass was added. The upper red dot indicates

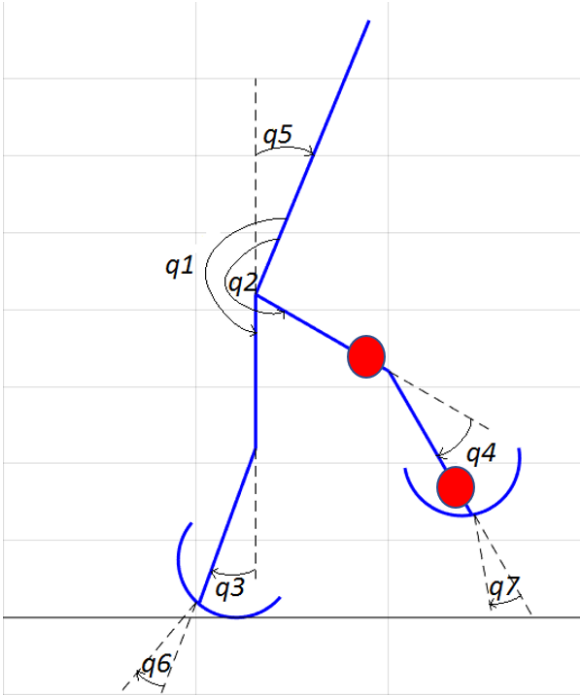

Figure 3.2: 7-link planar walker model with locations of added mass indicated by red dots.

a location just above the knee and the lower red dot indicates the location just above the ankle. The masses were added bilaterally and were symmetric on both the legs. We perform experiments for a walking speed of 1.3 m/s and the added mass ranges from 0 - 5 kg. We optimize the trajectory each time a mass is added so that the motion is optimal for the current mass distribution. Fig. 3.3 shows the effect of adding mass at the lower

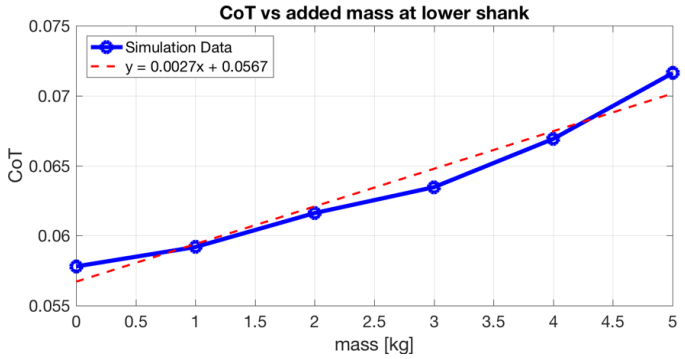

Figure 3.3: Variation of COT vs mass added at the lower shank.

shank (lower red dot in Fig. 3.2) on the energy required to carry it. We see that as the mass increases the cost of transport also increases significantly for mass added at lower shank suggesting that this might not be an appropriate location for adding mass. Fig.

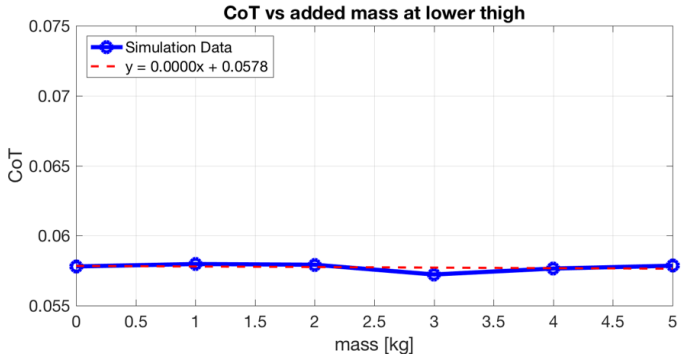

Figure 3.4: Variation of COT vs mass added at the lower thigh.

3.4 shows the effect of adding mass at the lower thigh on the energy required to carry it. Note that even though the change in cost of transport is relatively flat, the energy consumed is still increasing. This is because to calculate the COT, we normalize by the total weight of the system. As we see, the increase in energy is significantly smaller as compared to adding mass at lower shank.

In the second set of experiments, we add a constant mass and study the effect of increase in energy as the velocity of walking increases. Fig. 3.5 shows the increase in the cost of transport for increasing speeds of walking when a mass of 2.7 kg is added at

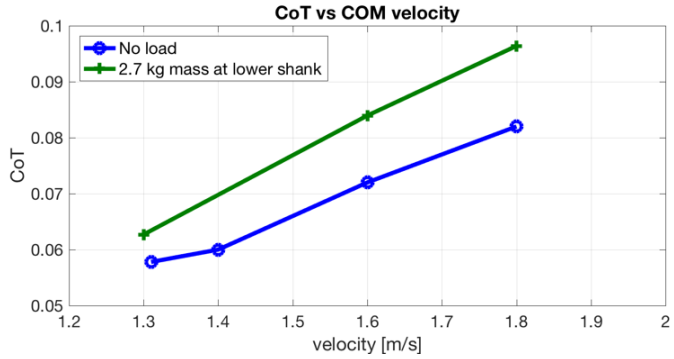

Figure 3.5: Variation of COT vs velocity of walking for a constant mass added at the lower shank.

the lower shank. The blue line shows the rate of change of COT vs. velocity when no mass is added on the model and the green line shows the rate of change for added load. Fig. 3.6 shows the change in COT for increasing speeds of walking. Here, the blue line

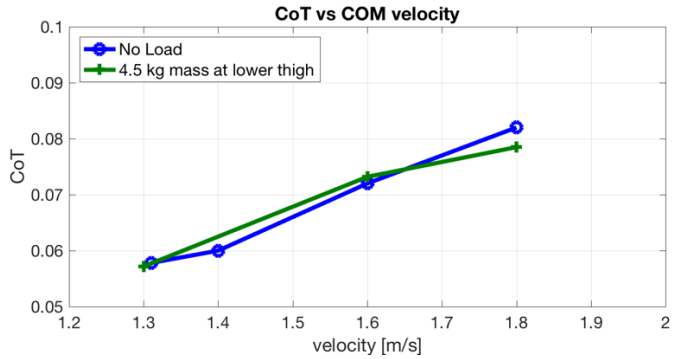

Figure 3.6: Variation of COT vs velocity of walking for a constant mass added at the lower thigh.

shows rate of change of COT when no load is added and the green line shows the rate of change when a load of 4.5 kg is added at the lower thigh. As we see, even though the added load is much higher for lower thigh the increase in COT is lesser as compared to that of lower shank.

Finally, we compare our simulation studies and actual human experiments in Fig. 3.7. The figure on the top corresponds to the simulation data and the figure on the bottom corresponds to the actual human experimental data. This figure shows the trend for a

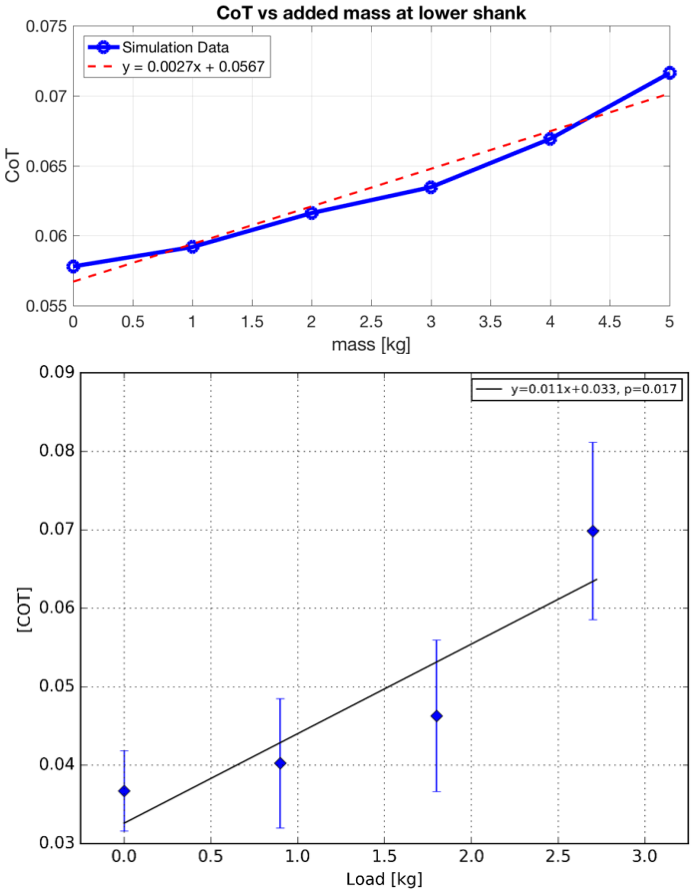

Figure 3.7: Variation of COT vs added mass at lower shank. The top figure shows the variation for simulation data and the bottom figure shows the variation for the actual human experimental data.

mass added at the lower shank. If we assume the muscle efficiency of around 25 percent [38], we would expect that the increase in cost of transport for actual human data should be 4 times that of the simulation data. On comparing the slopes in the two plots we find that the ratio comes out to about 4.03. This tells us that the trend observed in the human data is valid for analyzing the effect of added mass in the augmentation factor equation.

Both of our simulations show an increased impact of adding mass distally, near the ankle and foot. In contrast, the data also show a surprisingly low burden associated with a location just above the knee. These findings deserve further study and may significantly

influence design of future exoskeletons, particularly when power must be carried on board. Our studies have been designed to expand and refine the augmentation factor equation. Future revisions to the equation building upon this work should enable designers to quantitatively balance the power and mass of an exoskeleton more effectively. This understanding will allow exoskeleton designers to optimize performance more effectively, minimizing the arduous prototype and test cycle.

Chapter 4

Co-Dependence of Energy Optimality, Robustness and Design Parameters

Any practical locomotion needs to be both energy efficient as well as robust to any uncertainties in order to be viable. Underactuated locomotion is harder to control due to fewer actuators than degrees of freedom to control. At the same time, this underactuation also potentially reduces the amount of energy required. Hence, the trade-off between energy efficiency and robustness is very interesting for underactuated biped locomotion. This trade-off depends on a multitude of factors such as the design of the robot, the nature of the motion as well as the controller used on the robot. Each of these factors affects the robustness and energy efficiency of the system in unique ways and it is important to quantify such effects if we hope to understand the inherent nature of the trade-off between energy and robustness for underactuated biped locomotion. In this chapter, we demonstrate the importance of quantifying not only energy but also robustness as well as the significance of optimizing not only the trajectories but also feedback control policies

and the physical parameters of the robot.

4.1 Five Link Planar Biped Model

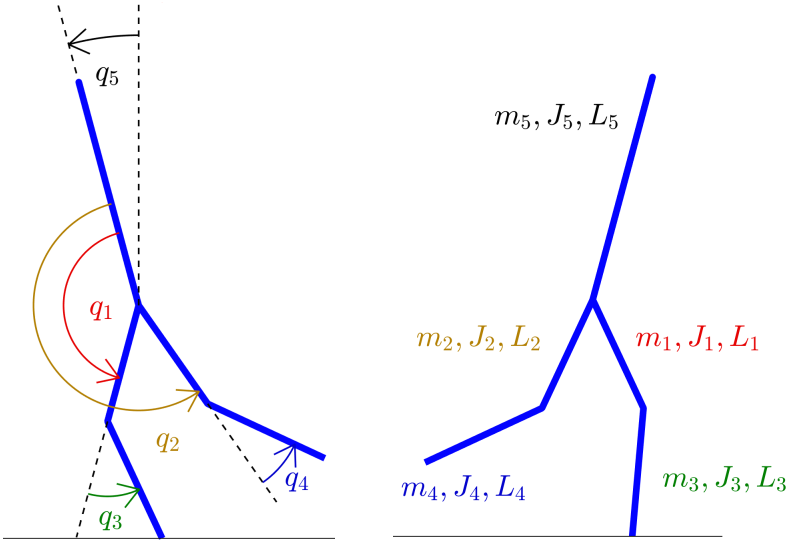

Figure 4.1: 5-link biped model. At left, q_5 is an absolute angle, measured with respect to vertical, while all other angles are relative. The lefthand image is drawn to clearly illustrate angles are positive in the counter-clockwise direction, throughout. At right, a typical pose, while walking to the right.

We use a 5-link model, shown in Fig. 4.1, for our simulations. The dynamics of this system are constrained to the sagittal plane only. We study several mass distributions, always enforcing that the total mass of the model is 70 (kg). The model has actuators at hips and knees, and the nonlinear dynamics can be written in the matrix form as

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu. \tag{4.1}$$

This well-studied model [8] has 5 degrees of freedom, corresponding to 5 joint angles given by $q := [q_1\ q_2\ q_3\ q_4\ q_5]^\mathsf{T}$, but due to the passive point-foot contact at the ground, the model still remains underactuated with $u \in \mathbb{R}^4$. We model the impact dynamics between

the swing leg and the ground as instantaneous and inelastic [39] to obtain joint velocities just after the impact. Since the impact model we use assumes inelastic collisions, some amount of energy is lost when the stance foot impacts the ground.

4.2 Trajectory Optimization

We discretize the dynamics and use direct collocation (DC) to generate trajectories for the walking motion as described in Chapter 2. Our approach is close to that suggested in [40], except that we use trapezoidal integration instead of backward Euler. Trapezoid rule integration can potentially result in lack of convergence. However, in our work, this had not been an issue, and results with trapezoid rule are significantly more accurate, when comparing the discretized (and thereby approximate) solutions inherent in this framework with subsequent high-resolution (1e-9) simulations of dynamics in Matlab.

The optimization problem is formulated as

$$\min_{q,\dot{q},u} \quad COT \tag{4.2}$$

$$\text{such that} \quad D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Hu \tag{4.3}$$

$$\Phi(q,\dot{q}) \leq 0 \tag{4.4}$$

where $q \in \mathbb{R}^n$ is the vector of generalized coordinates, $D(q) \in \mathbb{R}^{n\times n}$ is the mass inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ represents the Coriolis forces, $G \in \mathbb{R}^n$ contains the gravitational forces and $H \in \mathbb{R}^{n\times n-1}$ is the input (torque) mapping. $\Phi(q,\dot{q})$ is a vector of constraints. Constraints are imposed to make sure that the normal reaction at the point of contact with the ground is always positive. The optimization problem is set up such that at the end of the trajectory an impact at the ground happens. An additional constraint is added that the state of the model after the impact should match the initial condition

in order to obtain a limit cycle behavior. The objective function is the COT which is calculated as described in Chapter 2.

We implement this framework in Matlab making use of CasADi [41], which lets us calculate gradients for optimization using algorithmic differentiation to machine precision. (CasADi uses *C*omputer *a*lgebra *s*ystem syntax to perform *Algebraic Differentiation*; thus the name.) Using algebraic (and not numerical) differentiation greatly increases the stability and convergence properties of our optimization, while also reducing run time considerably.

Using CasADi to improve automated gradient calculation, the nonlinear programming (NLP) optimization itself is solved using IPOPT [33]. This choice (vs. use of SNOPT, Matlab's fmincon, etc.) is made based both on improved speed during our own in-lab testing experience and similar external benchmarking results [34].

The DC framework evaluates Eqs. 4.2-4.4 only at discrete time intervals, t_k, resulting in an approximation of the desired optimization problem. We use $\Delta t = h = 0.01$ (s) and integrate using the standard trapezoid rule. Also, we assume $u(t)$ is held via a zero-order hold for each time step (as opposed to a first-order hold). Our integration scheme is then

$$q_{k+1} = q_k + \frac{h}{2}[\dot{q}_k + \dot{q}_{k+1}] \tag{4.5}$$

$$\dot{q}_{k+1} = \dot{q}_k + \frac{h}{2}[f(q_k, \dot{q}_k, u_k) + f(q_{k+1}, \dot{q}_{k+1}, u_k)] \tag{4.6}$$

where $f(q, \dot{q}, u) = D(q)\backslash(-C(q, \dot{q}) - G(q) + Bu)$. We found that using a first order hold for input, i.e., replacing u_k with u_{k+1} at far right in (4.6) above, leads to trajectories that are undesirably oscillatory and not smooth. This problem is easily rectified by adding some level of regularization. However, the zero order hold method still converges more rapidly. Also important to note is that regularization increases the optimal cost by a small amount.

4.3 Motion Characteristics

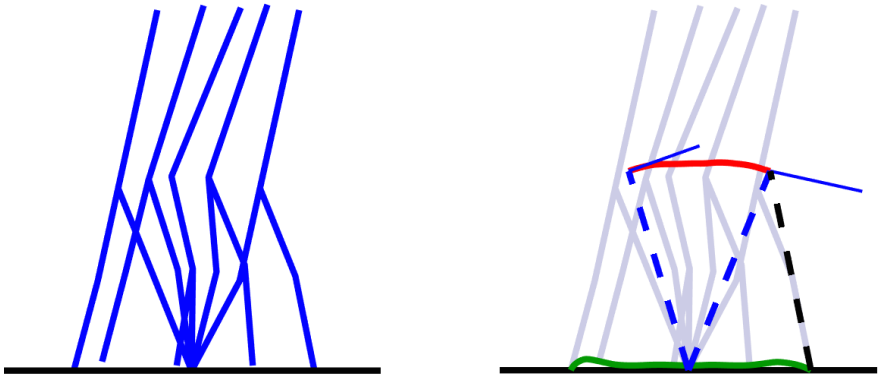

Figure 4.2: Typical motion for the 5-link walker, using trajectories generated from the optimization framework. The figure on the left shows snapshots of motion. On the right, trajectories of the COM and the end of the swing foot are overlaid.

Fig. 4.2 shows a typical motion generated for the mass distribution $m_5 = 50\ (kg)$, $m_1 = m_2 = 7\ (kg)$, $m_3 = m_4 = 3\ (kg)$, which corresponds roughly to a human mass distribution. Fig. 4.4 and Fig. 4.5 show the corresponding angle and angular velocity trajectories for that motion, and Fig. 4.6 shows the joint torques. All the results we present here are generated for a walking motion of stride length = 0.6 (m) over a time interval of 0.6 (s), resulting in a velocity of 1 (m/s).

A few details in these four figures are worth pointing out. First, note that all trajectories are divided into two subplots for better resolution and clarity, since q_1 and q_2 remain close to π, while the other joints are near 0. Upper plots correspond to upper leg segments; solid (blue) lines correspond to stance leg segments (femur and tibia).

Several characteristics seen in this example are common among optimizations we performed across a range of mass distributions, as itemized below:

1. The swing leg follows a very low trajectory, as depicted by the solid green line in Fig. 4.2. We enforce a minimum ground clearance of 2 (cm), except at the first and last 5 points of the trajectory, and the swing leg tip overshoots near the ends

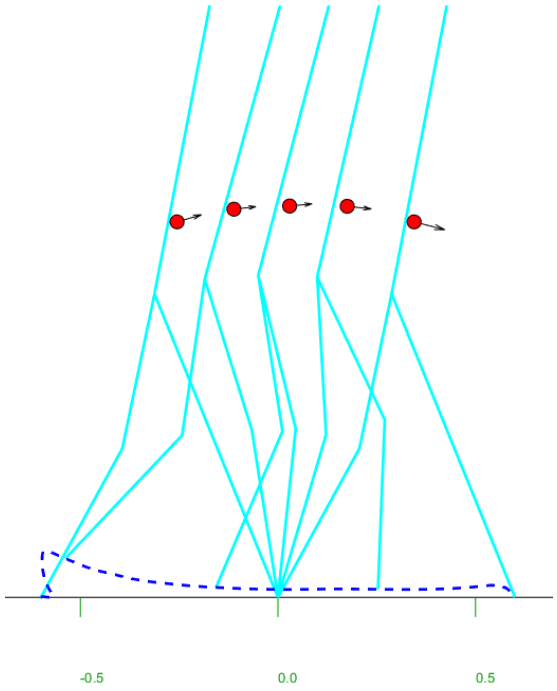

Figure 4.3: Detailed trajectory of the swing foot and the center of mass of the 5 link walker.

and grazes this value mid-gait. Without adequate ground clearance (e.g., 2 (cm)), resulting limit cycle gaits could not be stabilized.

2. The center of mass (COM) trajectory "flattens out" mid-stride, as opposed to following an arc, which is a feature also seen in human walking. This trajectory requires work in bending of the stance leg but results in less acceleration and deceleration vertically (against gravity), for lower accelerations overall of COM.

3. Also, rapid changes in velocities just before impact, as seen in Fig. 4.5, deflect the COM velocity at the end of gait slightly "upward", reducing kinetic energy losses at impact. The velocity vector, depicted as a solid blue line in Fig. 4.2 is close to orthogonal with the dashed line drawn from COM to stance leg tip at the start of the step (at 92.8°) and more obtuse at the end (100.0° w.r.t. current stance

contact, and 66.0° w.r.t. upcoming stance leg, as a black dashed line).

4. Also, the velocity vector is longer (i.e., faster speed) at the end, showing kinetic energy has built up, to compensate for dissipation at impact.

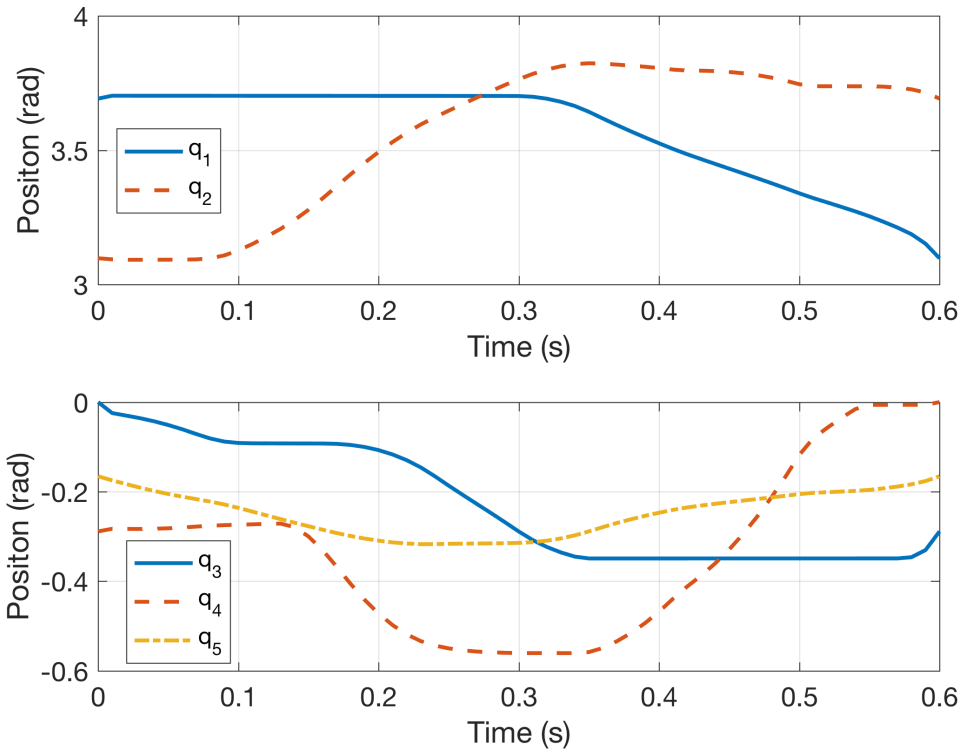

Figure 4.4: Angle trajectories for the walking shown in Fig. 4.2. The top plot shows the positions of the swing and the stance thighs, while the bottom plot shows the positions of the stance knee, swing knee and torso.

The velocity (Fig. 4.5) and torque (Fig. 4.6) trajectories also show repeatable features that were not anticipated but are (retrospectively) intuitively in agreement with our cost function, as noted in the rest of the list of features, below.

5. *Magnitude* of velocities of the stance leg segments (solid blue) increase *quite rapidly* at the very end of the step, in achieving the COM deflection upward (item 3).

The "toe-off" described above has a well-known benefit for energetics [42], but it also causes a problem:

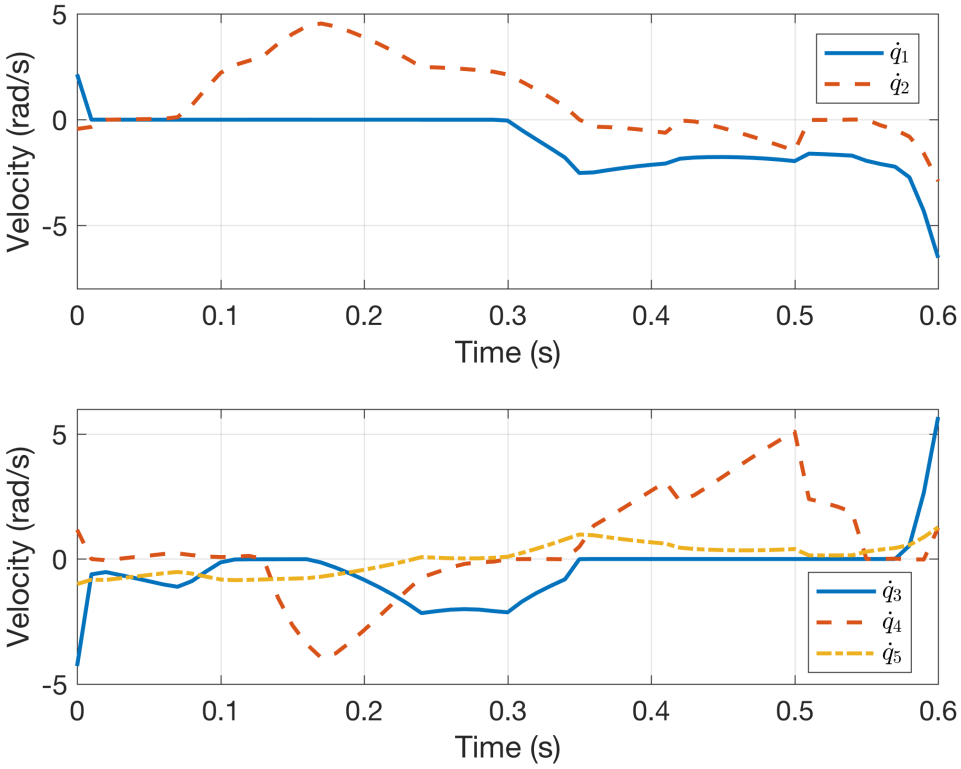

Figure 4.5: Velocity trajectories for the walking motion shown in Fig. 4.2. At top are velocities of the swing and the stance thigh; bottom plot shows the velocities of the stance knee, swing knee and torso

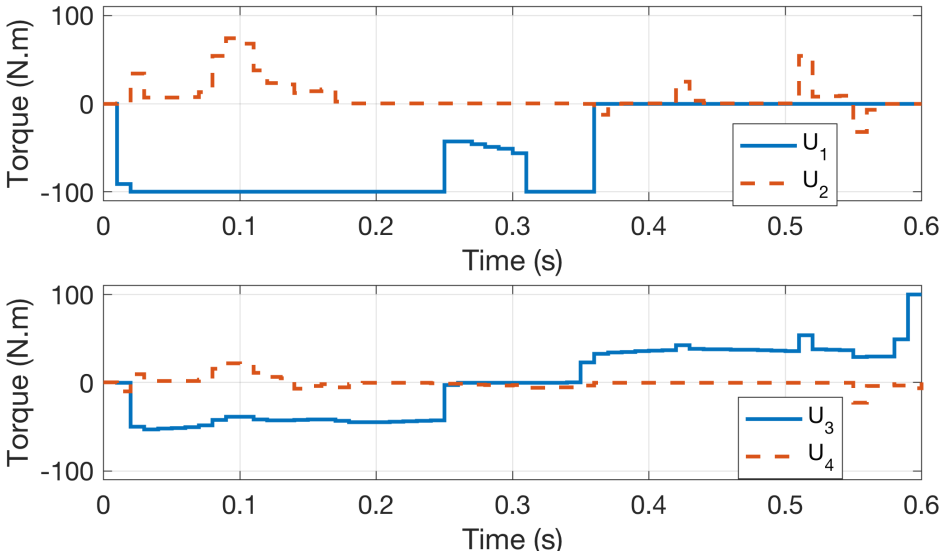

Figure 4.6: Torques for the walking motion in Fig. 4.2.

6. We observe that rapid increases in velocity near the end of a step, required for toe-off, also have an important (and unfortunate) negative effect on stability.

For example, when we attempted to use linear quadratic regulation (LQR) to provide feedback control, the resulting gait was not stable. Specifically, we linearized about each "knot point" of the optimal solution, and then controlled motions using the nominal feedforward torque (U_{ff}) added to feedback of the form $U_{fb} = -K(X - X_{nom})$, where $K = K(t)$ and $X_{nom} = X_{nom}(t)$ were interpolated between their values at the discrete points of the optimal solution. During continuous motion, the trajectories definitely converged toward the nominal trajectories as expected, but the effects through impacts were too destabilizing, resulting in falls after 3 to 6 steps.

This is a particularly interesting result, as it demonstrates evidence for a strong hypothesis that trajectories and feedback policies should be optimized as a concurrent problem, rather than planning ad hoc feedback subsequent to solving for a nominal trajectory. Finally,

7. values of u_n and the corresponding relative angular velocity $\dot{q}_n$ show a complementary behavior: when one is significant in magnitude, the other is near zero.

This makes sense, given $P_n = u_n \dot{q}_n$, from Eq. 2.9. In Figures 4.5 and 4.6, note in particular the solid blue lines, for the stance thigh (upper) and knee joints. The relative thigh angle (between torso and stance leg) is nearly zero while torque is at its maximum magnitude (-100 (N·m)), with the associated bobbing motion of the torso in Fig. 4.2 during the first 0.25 (s) of the step. For the stance knee, a period of negative velocity for $\dot{q}_3$ (i.e., knee bending) at mid-stance corresponds to a flat region in which $u_3 \approx 0$, i.e., bending almost passively during the gait. Near the end of the step, push-off with the stance knee is concentrated in particular at the last time step, perhaps exploiting the approximate nature of the optimization problem somewhat.

A different choice of cost function would result in somewhat different solution characteristics. We also tested a simple quadratic cost on control effort (i.e., cost = u^2); here, the "toe-off" behavior, with a spike in torque and velocity of the stance knee at the end of the step, also occurred. Overall, the trajectories are qualitatively more smooth for this cost function, however, and the overall COT is noticeably higher than when optimizing for COT specifically.

4.4 Effect of System Parameters on Energy

We repeated the same trajectory optimization to generate motions for our model on five different mass distributions. The physical parameters like link lengths and masses are shown in Tables 4.1 and Table 4.2, respectively. Each link is modeled as a simple rod, with COM at mid-length and the inertia is calculated accordingly.

Segment	$Length(m)$	$COM(m)$
$L_1 = L_2$	0.4	0.2
$L_3 = L_4$	0.43	0.215
L_5	0.77	0.385

Table 4.1: Length and COM parameters used in simulation experiments

Set	$m_1 = m_2(kg)$	$m_3 = m_4(kg)$	$m_5(kg)$	COT_{opt}
1	7	7	42	0.0992
2	5	7	46	0.0996
3	7	5	46	0.0861
4	5	5	50	0.0853
5	7	3	50	0.0705

Table 4.2: Mass parameters used in simulation experiments

Fig. 4.7 shows how changing ω_n affects COT. First, recall that discretized trajectory optimization solutions of true dynamics are always approximate ([5, 6, 16, 18, 20, 40, 30],

etc.), and also that ω_n is analogous to a "stiffness" (or proportional controller, "K_p"), within the PFL framework. Intuitively, increasing control gains would increase control effort, in following an arbitrary reference trajectory.

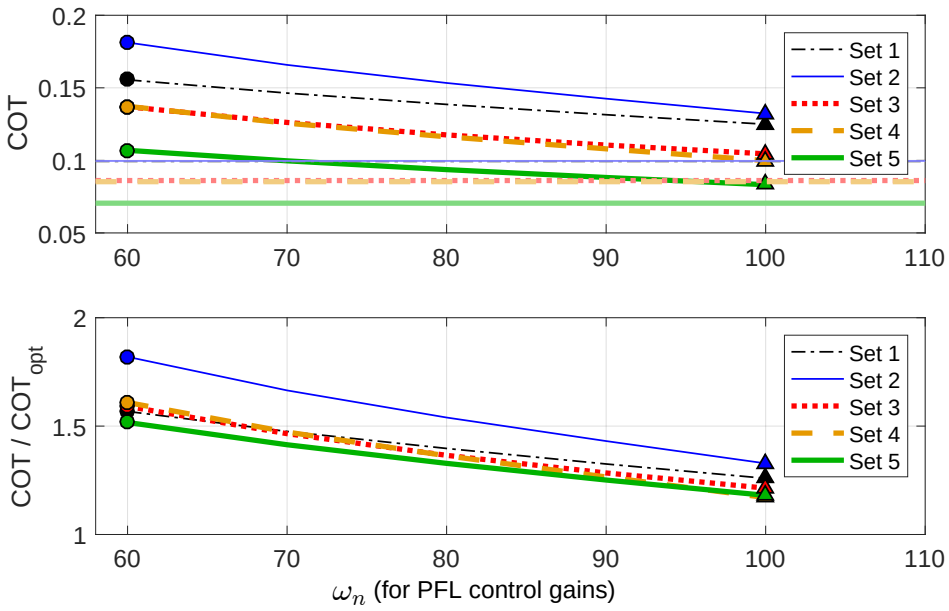

Figure 4.7: Cost of Transport (COT) as a function of ω_n.

However, our reference trajectories are far from arbitrary: they are precalculated to achieve a locally optimal (i.e., minimal) cost of transport – within some approximation errors. Therefore, increasing w_n monotonically decreases resulting COT in our higher-accuracy Matlab simulations. This suggests that the effect of following the trajectory is more significant than the increase in power due to higher peak torques.

Figure 4.7 both illustrates how COT converges exponentially downward as ω_n increases and also how different parameter sets result in a range of different errors, in comparing optimization results to more accurate simulations. For example, the lowest COT is for Set 5 from Table 4.2, which is also closest to a typical human mass distribution. The lower subplot of Fig. 4.7 shows the ratio of "actual" COT (from simulation) to the value output from optimization, also as a function of ω_n, where Set 5 also shows

the lowest approximation error.

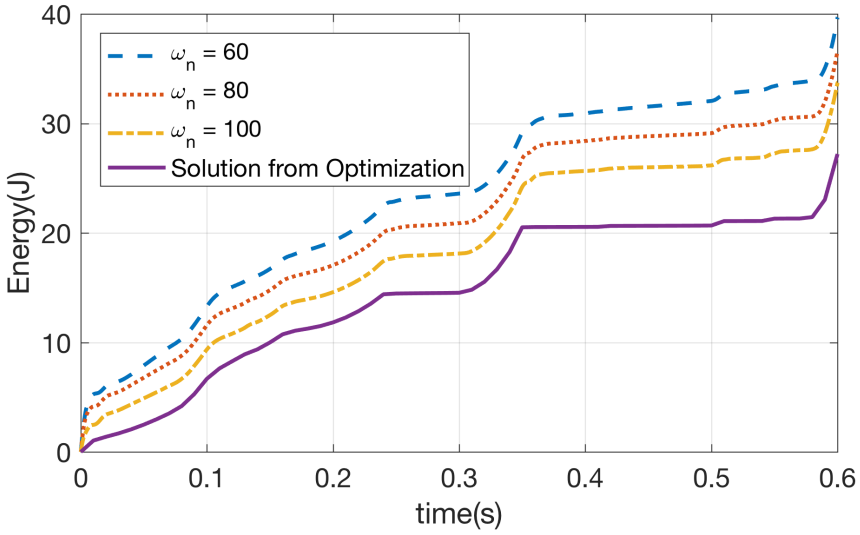

Figure 4.8: Variation of energy as ω_n changes.

Figure 4.8 shows the evolution of energy over time for different values of ω_n. The steep rise at the start is due to high peak torques as the controller tries to pull the actual trajectory back to the optimal trajectory.

Also, these and other simulations we have performed show that while increasing the mass of the lower leg (with constant upper-leg mass and total mass) results in a fairly linear increase in COT, as shown in the lower subplot of Figure 4.9, energy use remains close to flat as mass of the upper leg increases (while holding lower-leg and total mass constant). It is important to note that we optimize the gait for every mass distribution which gives us the optimal COT for the corresponding mass distribution. Corresponding trends relating mass distribution to stability are less apparent and deserve further investigation.

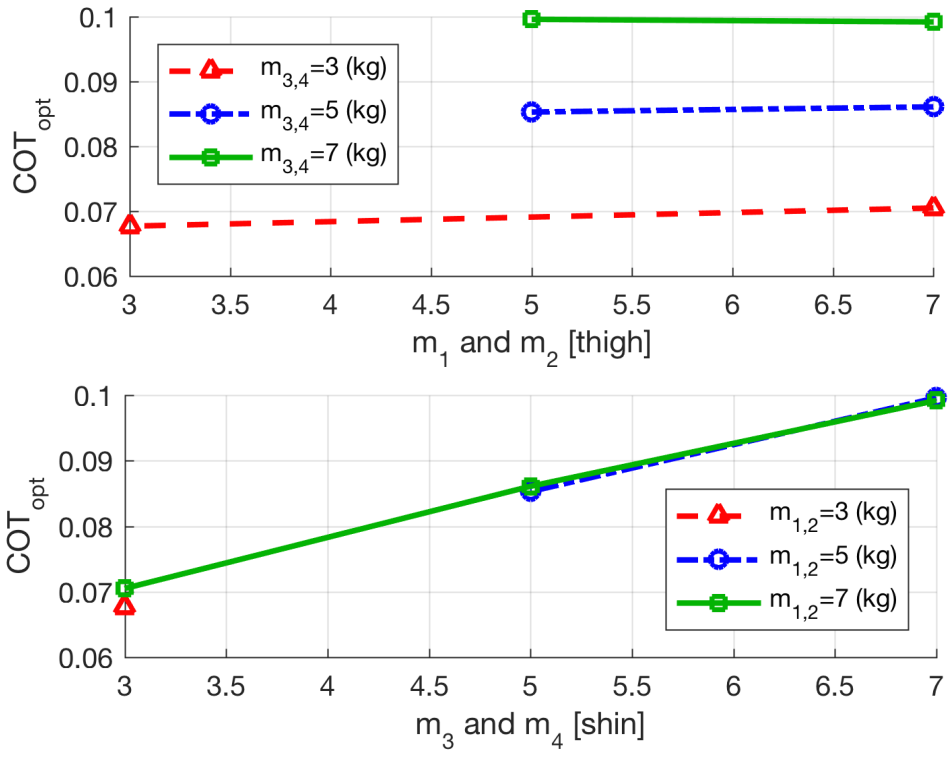

Figure 4.9: COT variations as upper or lower leg mass varies.

4.5 Effect of System Parameters on Stability

Fig. 4.10 shows the change in rate of convergence (i.e., largest/slowest discrete-time eigenvalue) as a function of ω_n when recovering from push perturbations. Three impulsive push tests were simulated for each set of parameters by applying an impulse of 10 (Ns)

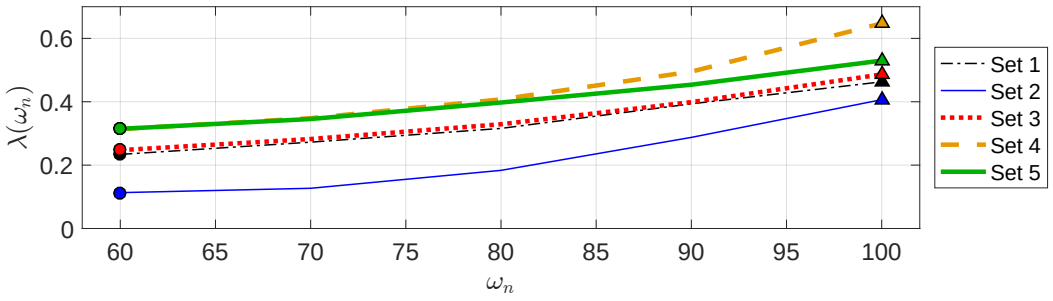

Figure 4.10: Rate of convergence, λ, as a function of ω_n, for push recovery.

(e.g., effectively a pulse of 1000 (N) for 0.01 (s)) at the hip, stance knee, or torso,

respectively. The velocities post impact were calculated as

$$D(q)\dot{q}^{+} - D(q)\dot{q}^{-} = E \cdot F_{ext}\Delta t \tag{4.7}$$

where $F_{ext}\Delta t = 10$ (Ns) is the applied pulse and $E = \frac{\partial p(q)}{\partial q}$ where $p(q)$ is the point at which force is applied. We calculate the convergence rate by simulating the model with optimal trajectory which has a limit cycle behavior and then introduce a disturbance as mention in Eq. 4.7. The disturbance causes the biped to deviate from the optimal trajectory. The low-level controller however, causes the model to converge to the limit cycle again asymptotically. We calculate the deviation of states on the Poincaré section and plot this error on a semilog scale as a function of number of steps taken. The slope of this plot is the convergence rate shown in Fig. 4.10.

Fig. 4.11 shows some of the error plots that we use to calculate the rate of convergence for the different mass and control parameters for disturbance introduced at one of the three locations: torso, hip and knee. The slope of the line on these plots gives us the rate of convergence back to to the limit cycle behavior. In some of the plots, we see that the dominant eigen-value changes as the model goes back to the limit cycle behavior as evident by two lines red and blue. In such cases we use the slower value as an estimate for the rate of convergence.

4.6 Energy Optimality and Robustness Trade-off

In the previous sections we study the effects of control and design parameters on the energy and stability of the systems. To explore their trade-offs, we presented COT vs rate of convergence for the different physical and control parameters. in Figure 4.12, which illustrates a Pareto frontier, formed essentially by sets 2, 3, and 5, while sets 1

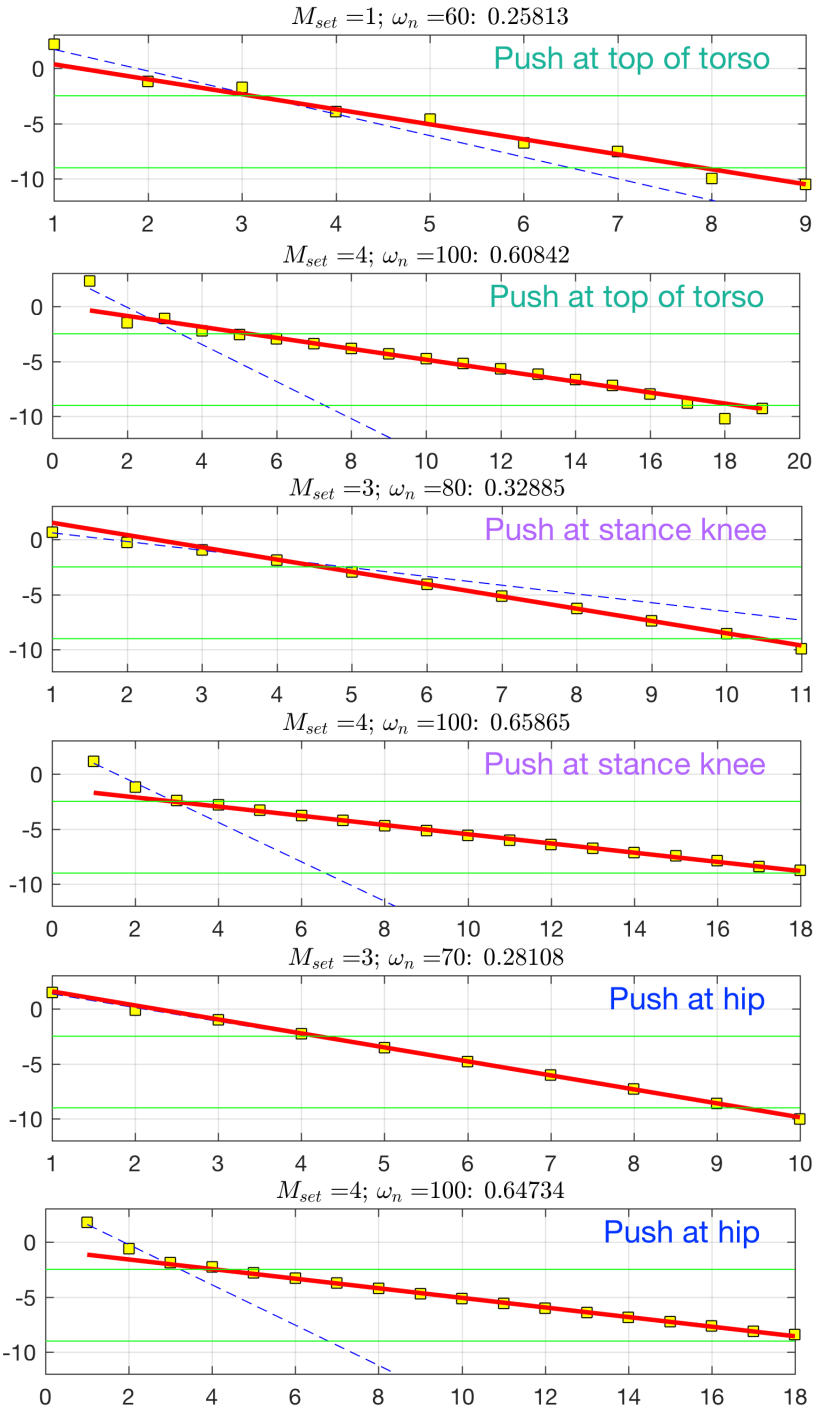

Figure 4.11: Error plots to calculate rate of convergence, λ, for different values of ω_n, for push recovery.

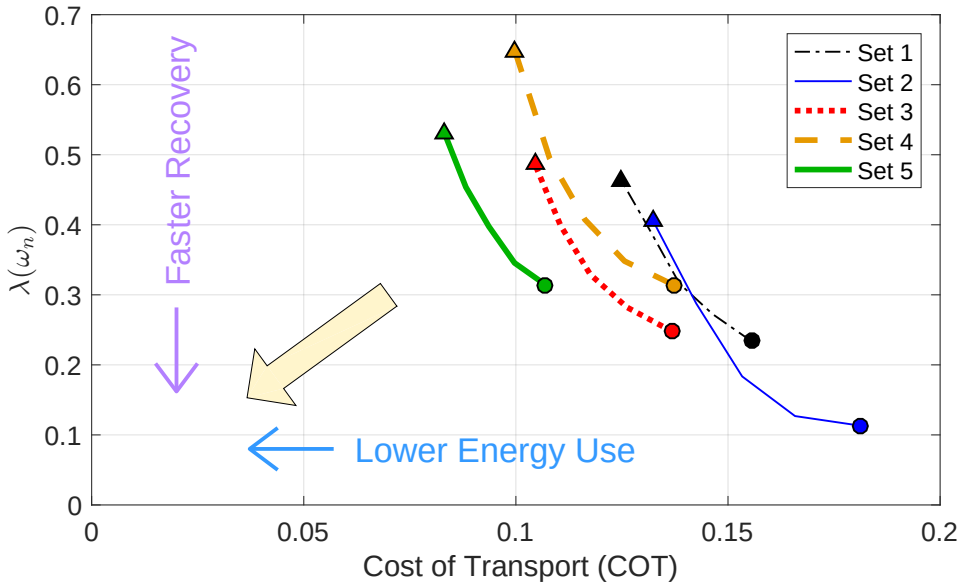

Figure 4.12: Cost of Transport vs rate of convergence $\lambda(\omega_n)$, for push recovery. Each line sweeps across results from $\omega_n = 60$ to $\omega_n = 100$.

and 4 provide poor trade-off characteristics by comparison. We hypothesize that a more comprehensive framework that includes energy, robustness, and physical parameters in a single optimization can improve the limits of such performance trade-offs, and developing such a framework is a goal for our future work.

4.7 Conclusion

Our simulation results demonstrate that although optimizing for energy alone and then stabilizing the trajectories can work, there is a need for a more cohesive framework that takes into account both energy as well as stability of the system for optimization simultaneously. Furthermore, because different physical robot design parameters have important effects on stability and energy properties, it would be prudent to include these parameters as open variables, via appropriate optimization frameworks.

Toward better understanding the effects of system parameters on the energy and sta-

bility of biped systems, we presented simulation data for 5 different sets of parameters. For the mass distribution sets we chose, the one which is most similar to human parameters (Set 5) does in fact have the lowest energy consumption. However, the set with the fastest rate of convergence is set 2. These phenomena may in part be a result of other factors, such as choice of feedback control structure and certainly warrant further study.

If we take a look at Figure 4.12, our goal in the future would be to build a cohesive framework that will take into account all these trends and push the performance of the system in the direction of the white arrow. In order for this to happen, the two challenges that need to be addressed are the quantification of robustness for biped locomotion and formulating a robustness metric that can be optimized simultaneously along with energy. We address a few of these challenges in the next two chapters.

Chapter 5

Metastable walking for push disturbances

In the last chapter we analyzed the effect of system parameters and feedback control policies on the energy efficiency and robustness of the five-link planar biped model. In this chapter we apply meshing tools to improve and analyze the performance of a five-link planar biped model to random push perturbations. Creating a mesh for a 14-dimensional state space would typically be infeasible. However, as we show in this chapter, low level controllers can restrict the reachable space of the system to a much lower dimensional manifold, which makes it possible to apply our tools to improve the performance. To validate the effectiveness of our tools in analyzing, quantifying and improving the performance of a system, we conduct simulations on two different sets of trajectories: one consisting of trajectories having only a single support phase, and a second set consisting of trajectories having both a double support as well as a single support phase. We use the concept of metastability as described in [14] to quantify the robustness of the 5 link planar biped model in the presence of push disturbances.

We perform experiments and analyze the performance for two different scenarios. For

the first case, our trajectory set is comprised only of single support phase trajectories. For the second case the trajectory set consists of trajectories which have both single support, as well as double support phase. All other parameters, except the trajectory and the structure of PFL, remain the same for both the experiments. The following sections detail each scenario.

5.1 Dynamic Model

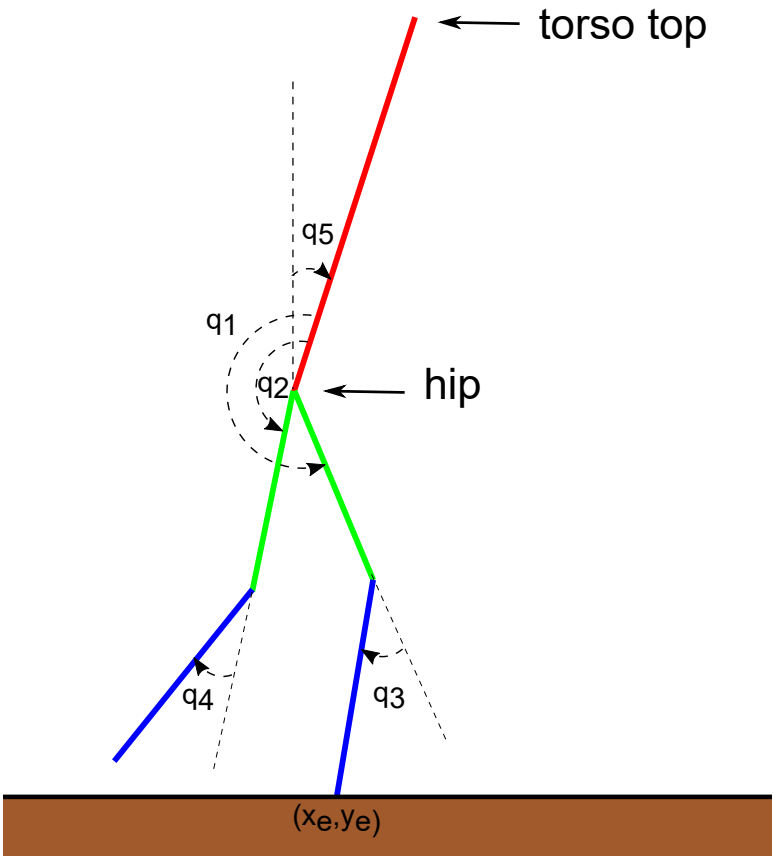

Figure 5.1: Model of the 5-link biped system used in simulations.

We use a point-footed, 5-link model as shown in Fig. 5.1 for our simulations. This model is studied extensively in [43]. We set the mass and length parameters such that the model has a total mass of 70 Kg and height of 1.6m. To calculate inertia values,

we consider the links to be thin rods with uniform mass distribution. Although [43] studies only single-support gaits, in which both feet are simultaneously on the ground only during instantaneous impacts, we use an extended model that allows us to design and simulate double-support trajectories, as well. Our model has 7 degrees of freedom: $q := [q_1\ q_2\ q_3\ q_4\ q_5\ xe\ ye]^T$, where q_1, q_2, q_3, q_4, q_5 are the five angles as shown, and xe and ye are the x and y coordinates of the end of the stance foot respectively. The simulation model can be compactly expressed in the following mathematical model:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + EF_{ext}. \tag{5.1}$$

Here, $D(q) \in \mathbb{R}^{7\times7}$ is the mass inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{7\times7}$ is the Coriolis matrix, $G \in \mathbb{R}^{7\times1}$ is the vector of gravitational forces, $B \in \mathbb{R}^{7\times4}$ is the input mapping and $E \in \mathbb{R}^{7\times4}$ is the mapping of forces acting on the stance leg and swing legs, respectively. During single support, the system is underactuated, as the input mapping B has more rows than columns. During double support, F_{ext} ensures no penetration of the leg into the ground and provides Coulumbic friction limits laterally. Sliding and/or bouncing of the point feet can occur in this model. Note that this differs from many formulations, which (artificially) treat such eventualities as failure modes in the dynamics, for simplicity. Swing leg impacts at the ground are modeled as instantaneous, inelastic collisions.

5.2 Trajectory Optimization Problem Formulation

Our optimization framework here is very similar to our work in the previous chapter with an important difference that we now have 7-DOF instead of 5 which allows the

model to have a wider range of motion such as jumping and sliding. We have

$$\min_{q,\dot{q},u,F_{ext}} \quad COT \tag{5.2}$$

$$\text{such that} \quad D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + EF_{ext} \tag{5.3}$$

$$\mathbf{\Phi}(q,\dot{q},u,F_{ext}) \leq 0. \tag{5.4}$$

Here, $\mathbf{\Phi}(q,\dot{q},u,F_{ext})$ is the vector of constraints and COT is given by Eq. 5.8. In addition, we also impose constraints on forces on the stance leg and swing legs. We generate trajectories assuming the coefficient of friction is 0.2, but we simulate the trajectories with a coefficient of friction of 0.5. A tighter bound on the friction is used because even though the controlled trajectory deviates from the optimal trajectory by a very small margin, it may cause slipping in certain cases. We use a Direct Collocation framework for optimization. For the purpose of trajectory optimization, we discretize the dynamics with a resolution of 0.01s and use the trapezoid rule for integration. For the inputs, we use a zero-order hold instead of first-order as this provided smoother results. We use IPOPT [33] for optimization, along with CasADi [31] to calculate the gradients for optimization using automatic differentiation. All the trajectories are designed for a fixed time length of 0.6s and are optimized for minimal energy use.

The cost function for our optimization is given by,

$$Cost = \int_0^t \sum_{n=1}^{7} |P_n(t)| dt, \tag{5.5}$$

where $P_n(t) = \tau_n(t)\omega_n(t)$. Here $\tau_n(t)$ is the torque and $\omega_n(t)$ is the angular velocity for the corresponding degree of freedom. We sum the absolute value of both the positive as well as the negative mechanical power. Since absolute value functions can be problematic for the convergence of optimization, we use the following approximation, where we have

also taken discretization into account:

$$Cost = \sum_{k=0}^{N-1} \sum_{n=1}^{7} \widetilde{P}_n(k)\Delta t. \tag{5.6}$$

Here

$$\widetilde{P}_n = \sqrt{P_n^2 + \epsilon}, \tag{5.7}$$

where ϵ is a smoothing factor. For our experiments, we used $\epsilon = 0.5$. Experimentally, smaller values of ϵ will work but also increase the time to find an optimal solution with not much difference in the final trajectories obtained. In addition to the cost, we also add some regularization terms to smooth the velocities and external forces. We found that smoothing the external forces was essential in order to control the double support phase trajectories on the model. Also of note is the fact that double support phase trajectories have a higher cost of transport as compared to single support phase trajectories. For the same stride length and ground clearance, the cost of transport for double support phase trajectories was 2-3 times that of single support phase trajectories. Detailed values are given in Table 5.1; all trajectories in this Table are for a constant time length of 0.6s. We calculate the Cost of Transport as

$$COT = \frac{Cost}{m_t g d}, \tag{5.8}$$

where $Cost$ is as described in Eq. 5.5, m_t is the total mass of the system, g is the acceleration due to gravity and d is the stride length.

Stride Length(m)	$COT(SS)$	$COT(DS)$
0.4	0.0501	0.1385
0.5	0.0809	0.1995
0.6	0.1332	0.3133

Table 5.1: COT for Single Support(SS) and Double Support(DS) trajectories, for different stride lengths.

5.3 Meshing for Metastable Systems

We record the state at the end of each step and use these discrete states (on a Poincaré section) to build a mesh $\mathbb{M}$, toward modeling stochastic, step-to-step dynamics. For our current work, we build and analyze a mesh that includes random disturbances, in the form of a push that can occur some time in the middle of the trajectory. Regardless of when (if at all) a push occurs, we record the subsequent post-impact state with the ground to create our mesh. We denote this post-impact state as $s = x^+$. The mesh also includes an absorbing failure state. All failure events transition to (and subsequently remain in) this state, in our discrete model of the dynamics. We then build the mesh, starting with each of the optimal limit cycles, along with the failure state. Non-failure states are simulated, using our level controller as described in Chapter 2, for each possible optimal trajectory found as described in the above section and for each of a finite set of possible push disturbances γ in our uncertainty set. That is, the control action is the choice of which reference trajectory, ξ, to use. Our meshing algorithm performs an exhaustive search, sequentially and deterministically exploring all possible states, s, that can be visited, to generate a mesh approximating the reachable state space of the modeled system, for the given control and disturbance sets. New states visited are added to the mesh if and only if they exceed a defined distance threshold from all previously created states in the mesh. More specifically, we use a simple threshold criterion, based

on a Euclidean distance metric given by

$$d(s_1, s_2) = \min_{s_2 \in \mathbb{M}} \sqrt{\sum_{i=1}^{n} (s_1(i) - s_2(i))^2}. \tag{5.9}$$

States are added to the mesh if $d(s_1, s_2) > d_{thr}$. When any part of the biped other than one or both of the point feet touches the ground, this is modeled as a transition to the failure state. Because we allow for sliding and bouncing in our simulation, we pick a threshold time limit of 0.4s after which the first impact of the swing foot with the ground terminates the current step.

Algorithm 1 outlines our meshing procedure.

A deterministic state transition matrix $T_{det}(s, \xi, \gamma)$ which contains the information of all state transitions is updated every iteration. In case of failure, the corresponding value in the $T_{det}(s, \xi, \gamma)$ is set to 1. When calculating the distance, we ignore the x position of the ground contact of the stance leg as it has no effect on the next step being taken.

For our uncertainty set, we create push scenarios, γ, with different magnitudes in forward as well as backward directions that occur at various times throughout the trajectory. For our current work, we restrict the maximum number of disturbances per simulation to 1. Some disturbances are applied at the hip and some occur at the top of the torso. A full profile of the disturbance set is shown in Fig. 5.2. Our goal is to "push" the trajectories during walking away from their nominal paths in state space, to explore the mixing effects over time of cumulative disturbances that vary in direction, magnitude, and timing during the gait. Our underlying hypothesis is that while the system does not converge to any particular fixed point under such conditions, it is also limited to a relatively low-dimensional subset of the full state space, making long-term failure probabilities calculable.

Algorithm 1 Meshing Algorithm

Input: Initial matrix of states S, containing two states: s_1 (failure) and s_2 (one initial, non-failure state).
Disturbance profile D.
Distance threshold d_{tr}.
Set of controllers Z.

Output: Final matrix of all states S, discretely spanning the reachable state space.
Deterministic state transition matrix $T_{det}(s, \xi, \gamma)$.

Initialization:
Current state index: cur = 2
Total number of states: nstate = 2
while cur $\leq$ nstate **do**
  xcur $\leftarrow$ S(cur)
  for all controllers ξ in Z **do**
    for all disturbances γ in D **do**
      x, flag $\leftarrow$ simulatedynamics(xcur, ξ, γ)
      if flag = 0 (Step taken successfully) **then**
        if d(x, s) $> d_{tr}$ $\forall$ s $\in$ S **then**
          add x to S and set $T_{det}(xcur, \xi, \gamma)$ = nstate
          nstate $\leftarrow$ nstate + 1
        else
          set $T_{det}(xcur, \xi, \gamma)$ = index of s for d(xcur, s) $< d_{tr}$
        end if
      else
        set $T_{det}(xcur, \xi, \gamma) = 1$ (index to failure state)
      end if
    end for
  end for
  cur $\leftarrow$ cur + 1
end while
return S, T_{det}

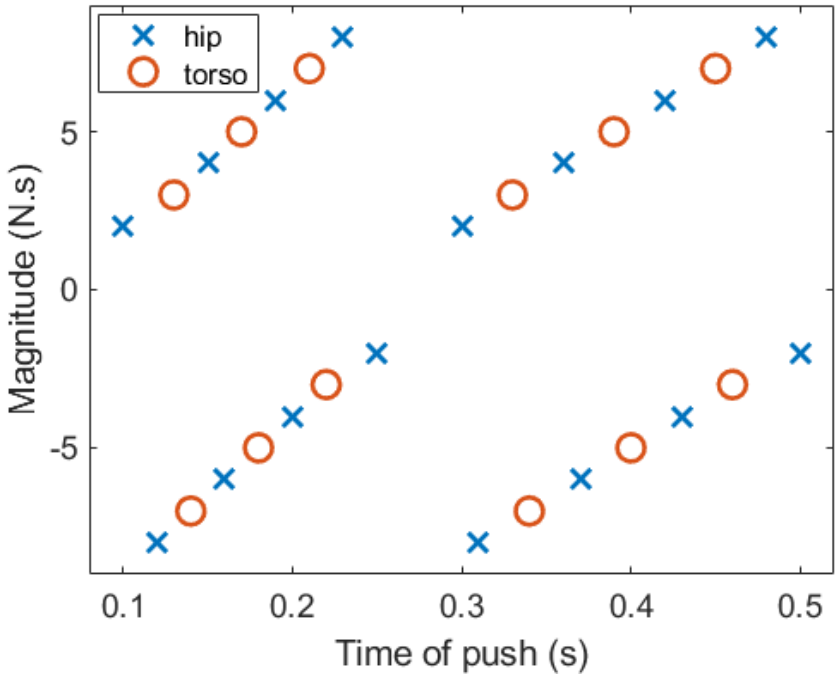

Figure 5.2: Disturbance profile used in the simulations. Disturbance is applied at two different locations namely the hip and the top of the torso.

5.3.1 Stochastic State Transition matrix

Once the mesh is complete and we have a deterministic state transition matrix as described above, we proceed with calculating the optimal policy for the mesh, which depends on the assumed probability distribution of the disturbances. The optimal policy given by $\pi(s)$ is a function of the state. Once we have this policy, the controller for the next step is selected by $\xi = \pi(s)$. We study a range of values for the probability distribution, $P(\gamma)$, for our uncertainty set. For each, we use value iteration to converge to a stable policy, where the value of being in state i is given by

$$V(i) := \max_{\xi}(\sum_{\gamma} P(\gamma)(R_j + \alpha V(j))), \tag{5.10}$$

and the optimal policy is given by

$$\pi(i) := \underset{\xi}{\text{argmax}}(\sum_{\gamma} P(\gamma)(R_j + \alpha V(j))), \tag{5.11}$$

where α is the learning rate, which we set to 0.99 for our experiments, and R_j is the reward function which is given by

$$R_j = \begin{cases} 0, & \text{if } j \text{ is } 1 \\ 1, & \text{else.} \end{cases} \tag{5.12}$$

This function rewards the control action so long as it does not lead to a failure state. Using this optimal policy, we calculate the stochastic state transition matrix:

$$T(i,j) := \sum_{\gamma} P(\gamma) f_j \tag{5.13}$$

where

$$f_j = \begin{cases} 1, & \text{if } T_{det}(i, \pi(i), \gamma) = j \\ 0, & \text{else.} \end{cases} \tag{5.14}$$

That is, the action depends deterministically on the present state, each particular disturbance then maps deterministically to one particular future state under this control, and the probability mass function for disturbances sets the relative weights in each possible arrival state. Note, each row in the stochastic transition matrix correspondingly sums to one.

5.3.2 Mean First Passage Time

The concept of mean first passage time (MFPT) for metastable walking systems is explained in detail in [44] and [14]. We highlight important results here that we will be using in the rest of the paper; they focus on an eigen analysis of the *transpose* of the stochastic transition matrix, T. First, we define a metastable distribution, ϕ, as the stationary distribution in state space, conditioned on not having entered the failure state:

$$\phi_i = \lim_{n\to\infty} Pr(X(n) = x_i | X(n) \neq x_1). \tag{5.15}$$

The mean first passage vector m where m_i is the mean time for the state i to go into failure is given by

$$m_i = \begin{cases} 0, & \text{if } i = 1 \\ 1 + \sum_{j>1} T_{ij} m_j, & \text{else.} \end{cases} \tag{5.16}$$

Vector m can be calculated by

$$m = \begin{bmatrix} 0 \\ (I - \hat{T})^{-1}\mathbf{1} \end{bmatrix} \tag{5.17}$$

where $\hat{T}$ is T with the first column and row (corresponding to the failure state, for which $m(1) = 0$) removed. This in turn lets us calculate the system wide MFPT:

$$M = \sum_i \phi_i m_i. \tag{5.18}$$

For a more efficient calculation of M, we use the eigenvalues of the stochastic transition matrix. Because of the structure of T, with $T(1,1) = 1$ corresponding to the absorbing failure state, the largest eigenvalue has the value 1: failure is a persistent state. Let λ_2

be the second largest eigenvalue of T. λ_2 denotes the probability of taking a successful step, assuming initial conditions have been forgotten and the walker is in a non-failure state; i.e., the probability of failure is $1 - \lambda_2$. The probability of taking only n steps is then

$$Pr(X(n) = x_1, X(n-1) \neq x_1) = \lambda_2^{n-1}(1 - \lambda_2). \tag{5.19}$$

By calculating the expectation of this probability distribution, we obtain the approximate MFPT for the system given by

$$M = \frac{1}{(1 - \lambda_2)}. \tag{5.20}$$

Finally, note that normalizing all the non-failure-state elements in the eigenvector associated with the 2^{nd}-largest eigenvalue of T^T (the *transpose* of T) correspondingly yields the metastable distribution, ϕ, in Eq. 5.15.

5.3.3 Average Cost of Transport

We can calculate the average COT for the system, based on the metastable distribution ϕ. To do this, we first calculate the average COT for each state in the mesh using

$$\overline{COT}_i = \sum_{\gamma} COT(i, \pi(i), \gamma) P(\gamma) \tag{5.21}$$

where we assume $COT(i, \pi(i), \gamma)P(\gamma) = 0$ if $T_{det}(i, \pi(i), \gamma) = 1$, i.e., if state i transitions to the failure state. The average COT for the whole system is then

$$\overline{COT} = \sum_{i} \overline{COT}_i \phi_i. \tag{5.22}$$

5.4 Dimensionality Analysis

As the dimensionality of a system increases, discretized (e.g., mesh-based) machine learning methods become impractical, due to Bellman's so-called "curse of dimensionality". For a D dimensional system, to make the state-state transition mappings more accurate we can reduce the distance between mesh points. The dimensionality of the system can be used to predict the resulting change in the mesh size. More specifically, if we wish to change the distance between mesh points from s to s/k, then we would expect the number of mesh points to scale as $N \propto k^D$. Figure 5.3(a) shows some simple examples of this. For example, for a 3D cuboid ($D = 3$) if we reduce the side length (s) of each element in a uniform grid to 1/2 of its original side, there will be $N = 2^3 = 8$ grid elements.

For some systems, this scaling is not an integer, due to some inherent structure of a system within state space. Fractals are a common example of this. The Koch curve, also known as the Koch snowflake, is illustrated in Fig. 5.3(b). To generate the shape, start with a single side (or an equilateral triangle, for a true "snowflake"). Then, divide each side into 4 new sides, each 1/3 of the previous side length; repeat forever, and you have a Koch curve.

The Koch snowflake has a dimensionality of about 1.26. To have a better understanding, imagine we wish to select a non-uniform set of points to approximate the full set. Paralleling the meshing approach used in this paper, we first pick a particular "distance threshold", d_{thr}, to compare candidate new mesh points with existing one, as in Equation 5.9. New candidate points are found and added to the mesh if and only if they are outside a "bubble" of radius d_{thr} of all the existing mesh points, resulting (after an exhaustive search) in a non-uniform mesh. Figure 5.3(c) shows a resulting example. Since $N \propto k^D$, and d_{thr} is in turn proportional to $1/k$, $D = log(N)/log(k) =$

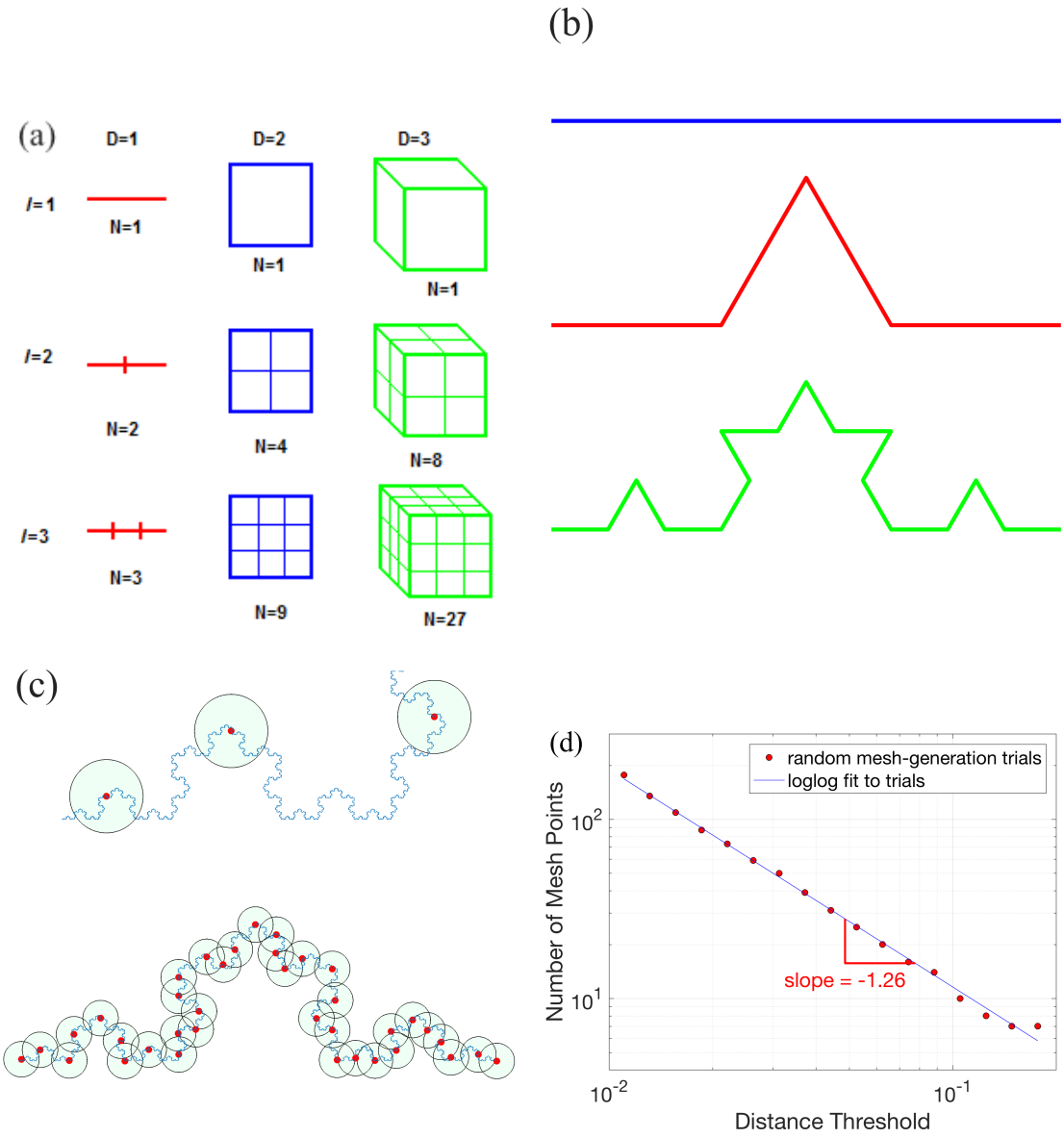

Figure 5.3: Fractional Dimensionality. (a) Uniform meshing examples [3], (b) Koch snowflake segment, (c) Non-uniform meshing example (see text for details), (d) slope of a loglog plot of "Distance Threshold" versus "Number of Mesh Points" is -1.26, which is correspondingly an estimate for the negative of the dimensionality of the Koch snowflake.

$log(N)/(-log(d_{thr}))$, and the negative slope of a log-log plot, such as shown in Fig. 5.3(d), gives the dimensionality, D. We use this method in the following sections to estimate mesh dimensionality, which describes scaling of the number of discrete mesh points as

d_{thr} varies.

5.5 Single Support Phase Walking

We use 3 trajectories $\{\xi_1, \xi_2, \xi_3\}$ with stride lengths of 0.4m, 0.5m and 0.6m respectively in our trajectory set $\mathbb{Z}$ to generate the mesh. All these trajectories have a time length of 0.6s and have been optimized for energy. We then use PFL as stated above to control these trajectories. For the trajectories in this set, matrix S as given in Eq. 2.12 sets the four acceleration terms that are directly controlled. Fig. 5.4 shows one example gait trajectory. Because of the sensitive nature of the inelastic collision at the end of each

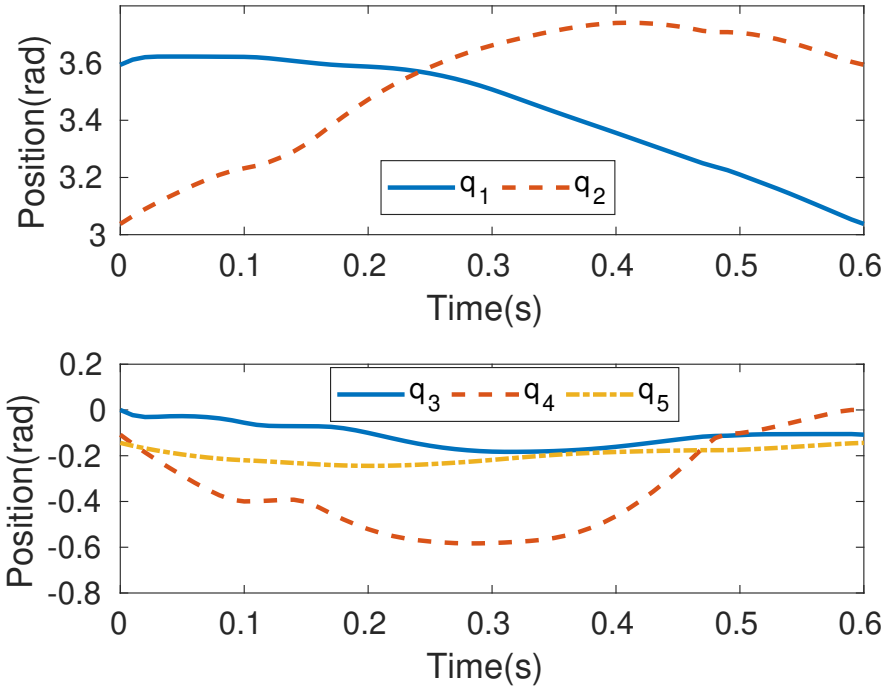

Figure 5.4: Plot of position trajectories for a 0.5m stride length trajectory having only single support phase

footstep, COT of controlled gaits using a smaller integration time step for more accuracy deviates from the value predicted through more coarse integration during trajectory

optimization. Table 5.2 shows this deviation. Table 5.3 shows the number of states for

Stride Length(m)	Predicted COT	Controlled COT
0.4	0.0501	0.1507
0.5	0.0809	0.1021
0.6	0.1332	0.1417

Table 5.2: COT for single support phase trajectories.

different meshes that we generate. As expected, the number of states increase by an exponential factor as d_{thr} decreases. If we plot the data in this table on a logarithmic scale,

d_{thr}	0.3	0.4	0.45	0.5	0.6
# of points	6089	2333	1856	946	779

Table 5.3: Number of points for different threshold values for mesh generated with single support phase trajectories

the slope gives the dimensionality by which the mesh grows. Fig. 5.5 shows the said plot. For the case of trajectories with only single support phase, the dimensionality comes out to be approximately $n \approx 3.1$ which is much less than the total 14 dimensional state space. It is this low dimensionality that allows us to use our meshing techniques even for real world noise scenarios like push disturbances. Once we have the mesh, we can calculate the MFPT and other important metrics. The following results are obtained for a mesh with $d_{thr} = 0.6$, along with the following probability distribution for the uncertainty set.

$$P(\gamma) = \begin{cases} 0.8, & \text{if no disturbance} \\ 0.2/28, & \text{else.} \end{cases} \tag{5.23}$$

Our uncertainty set $\mathbb{U}$ consists of 29 scenarios which are shown in Fig. 5.6. Of the 29 cases, 1 case is a no-disturbance scenario.

Using Eq. 5.20, the MFPT for our mesh comes out to 103.25, suggesting that with our optimal policy and the disturbance probability given, the robot would take approximately

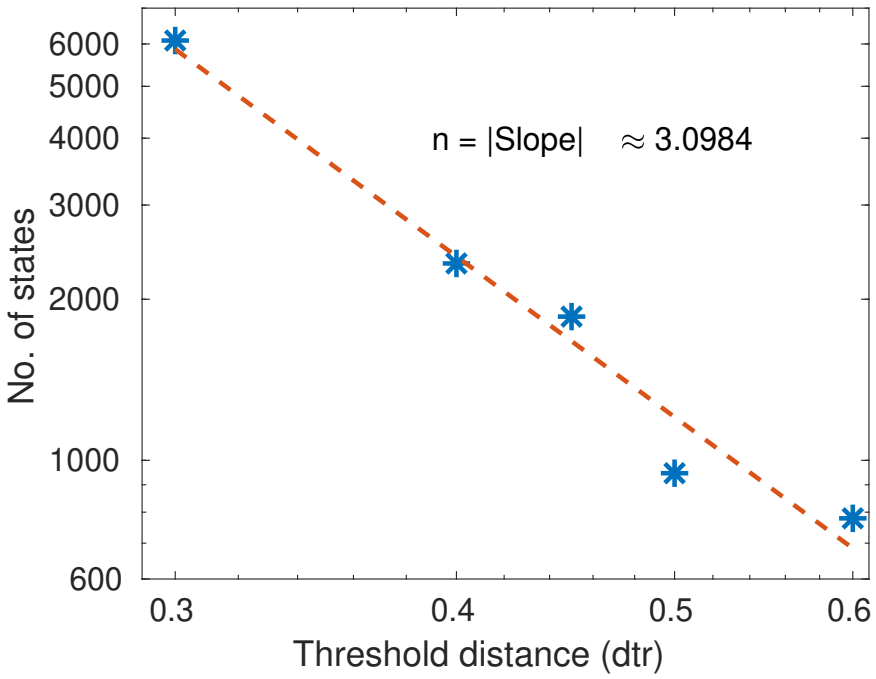

Figure 5.5: Dimensionality of mesh growth for Single Support trajectories.

103 steps before falling. From Eq. 5.22, we get the average cost of transport for the metastable distribution of the system to be 0.2751. Using our tools, we can perform various interesting analyses, such as calculating the sensitivity to disturbances. To do so, we use the following probability distribution. Here, γ_i is the disturbance for which we are calculating the sensitivity.

$$P(\gamma) = \begin{cases} 0.7, & \text{if no disturbance} \\ 0.2, & \text{if } \gamma = \gamma_i \\ 0.1/27, & \text{else.} \end{cases} \tag{5.24}$$

We then proceed to calculate the MFPT for the given probability distribution. We do this for all the disturbances in $\mathbb{U}$ and the results are as shown in Fig. 5.6.

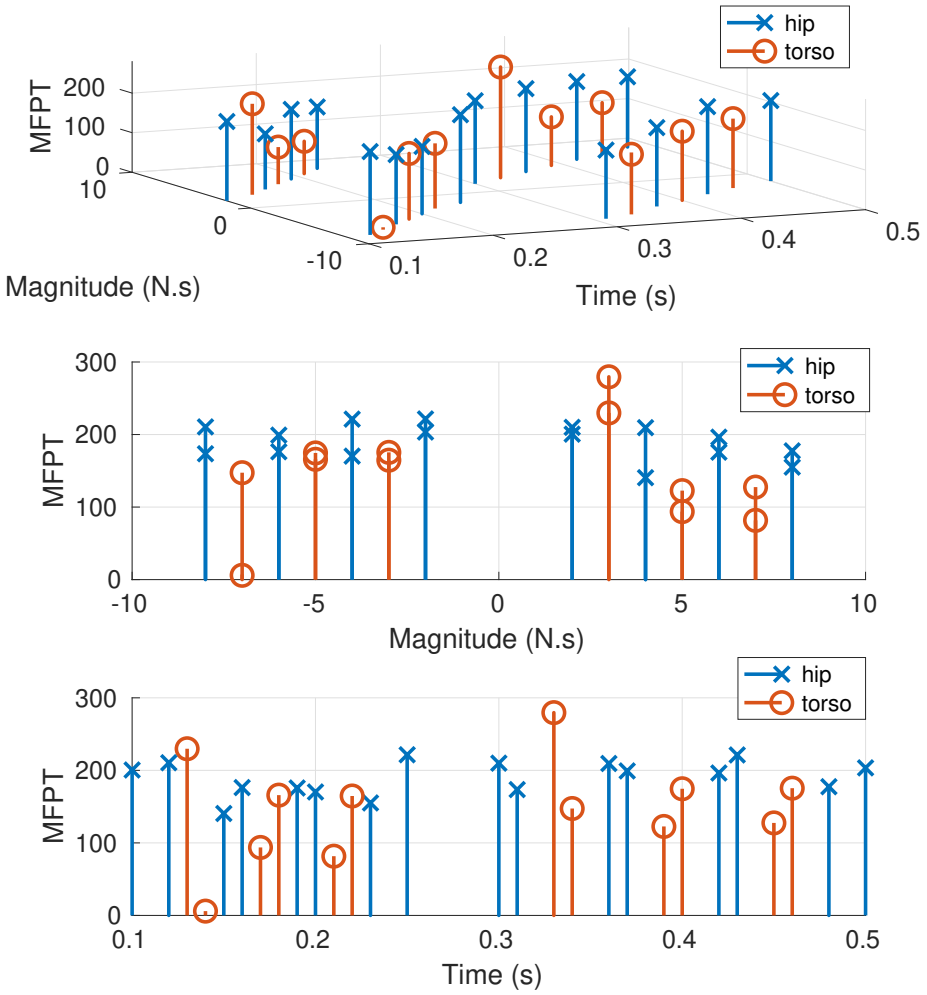

Figure 5.6: Plot showing the MFPT for various disturbance probabilities. The top figure shows the MFPT as a function of both the magnitude as well as the time of disturbance. The middle and bottom plot show the MFPT vs magnitude and time respectively.

5.6 Double Support Phase Walking

As in the previous section, we use 3 trajectories with a stride length of 0.4m, 0.5m and 0.6m to generate our mesh. These trajectories are also optimized for energy and have a time length of 0.6s. These trajectories differ from the trajectories of the previous

section in that they have a double support phase for 20% of the gait duration. The gait starts in the double support phase and ends at the end of the single support phase with impact to the ground. These post-impact states are added to the mesh. Fig. 5.7 shows one such trajectory. To control these trajectories, we use the same PFL structure but

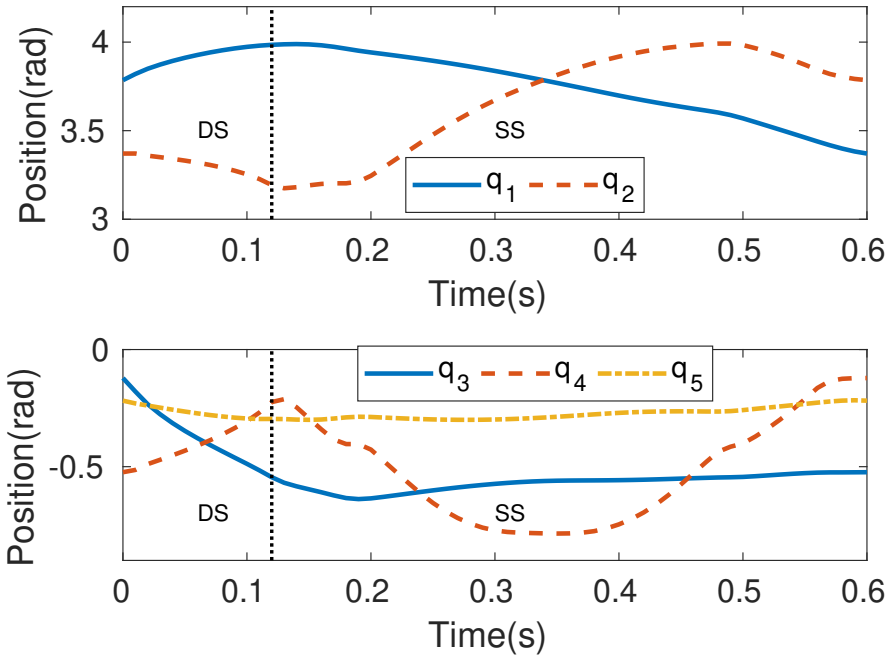

Figure 5.7: Plot of position trajectories for a 0.5m stride length trajectory having both double support (DS) and single support (SS) phase. The transition from double support to single support happens at 0.12s mark.

with a switching scheme. We divide the trajectory into 3 different time intervals 0-0.15s, 0.15-0.53s and 0.53-0.6s. For time intervals 1 and 3 we set

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.25}$$

and for time interval 2 we set S as given in Eq. 2.12. We introduced the switching scheme because the PFL scheme used in the previous section was not able to stabilize the trajectories having both a double support and a single support phase. The choice of the time intervals was based on the transition from double support to single support phase and some experimental simulations.

Tables 5.4 and 5.5 show the COT for the trajectories and the number of points for mesh generated with different d_{thr} values respectively.

Stride Length(m)	COT	Stable COT
0.4	0.1385	0.2216
0.5	0.1995	0.2733
0.6	0.3133	0.3618

Table 5.4: COT for trajectories having both double support and single support phase before and after PFL is applied.

d_{thr}	0.4	0.45	0.5	0.6
# of points	7260	5645	3925	2244

Table 5.5: Number of points for different threshold values for, trajectories having double as well as single support phase

Fig. 5.8 shows the plot to calculate the dimensionality factor for the double support phase trajectories. The MFPT for the case with $d_{thr} = 0.6$ for the probability distribution given in Eq. 5.23 for these set of trajectories comes out to be 496.88 with an average cost of transport $\overline{COT}$ of 0.4993. This is significantly more robust than that for trajectories having only single support phase but at the same time there is a trade off in terms of the amount of energy expended. This may also be in part due to the different PFL schemes used in both cases, however, the fact that the dimensionality in both cases comes out nearly the same indicates that the significant performance improvement results not from the contracting effects of the low level controllers but mostly due to the fact that double

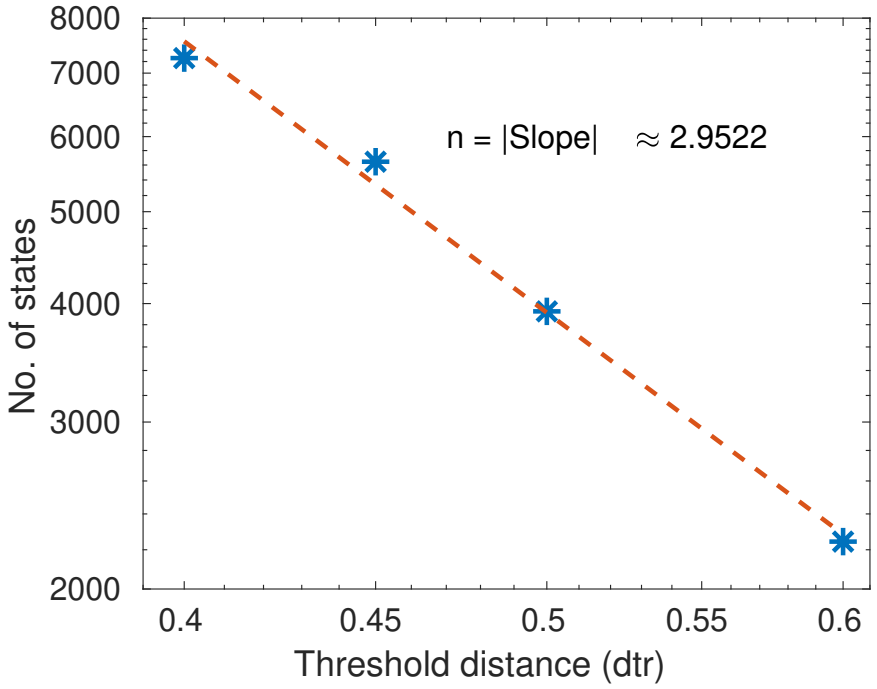

Figure 5.8: Dimensionality of mesh growth for trajectories having both double support and single support phase.

support phase introduces an over actuated phase in the simulation where we can control all the degrees of freedom of the system.

We also build a mesh by combining the trajectories from both sections above. As expected, this mesh provides a superior performance compared to either of the meshes generated in the previous section. For a $d_{thr} = 0.6$ this mesh has 10,603 states with a MFPT of 2,441.58 for a probability distribution given in Eq. 5.23. This is expected because more control policies (trajectories) allows for a higher probability of taking a successful step after a disturbance has been introduced. We also see that this mesh has a significantly higher number of states as compared to the mesh with just single support phase or double support phase trajectories. This may be due the mixing effect of the control policies leading to the exploration of previously unexplored states.

5.7 Sensitivity Analysis

An obvious question that comes to mind is how robust the policy is to change in the probabilities of disturbances so we analyze the sensitivity of the policy. For this, we pick a probability value for each disturbance from a uniform distribution between 0 and 1. We then normalize these probabilities such that the probabilities of all disturbances sum to x. The probability for no push scenario is then $1-x$. For each x we collect 10 data points. Fig. 5.9 shows how much the performance is affected due to change in the probability of the disturbance. The percent policy change gives the percentage of states that have a different policy when using the optimal policy versus the standard policy. The standard policy in this case is the optimal policy generated for the probability distribution given by Eq. 5.23.

As we see, the optimal policy is quite robust to the changes in the probability distribution of the noise. The top figure shows that the maximum deviation in the policy for our experiments is about 3 percent. This means that only 3% of the states have a different control action than when using our base policy.

5.8 Conclusion

Our experiments have shown that our tools previously developed for improving and analyzing the performance for waking on rough terrain are also applicable to more random real world noise scenarios like push disturbances. In addition, our simulations also show that trajectories that have a double support phase are much more robust to pushes as compared to trajectories with only single support phase. This result is intuitive, but our tools allow us to quantify such differences in long-term performance. Our tools also allow us to do sensitivity analyses for different kinds of disturbances. The plots clearly indicate

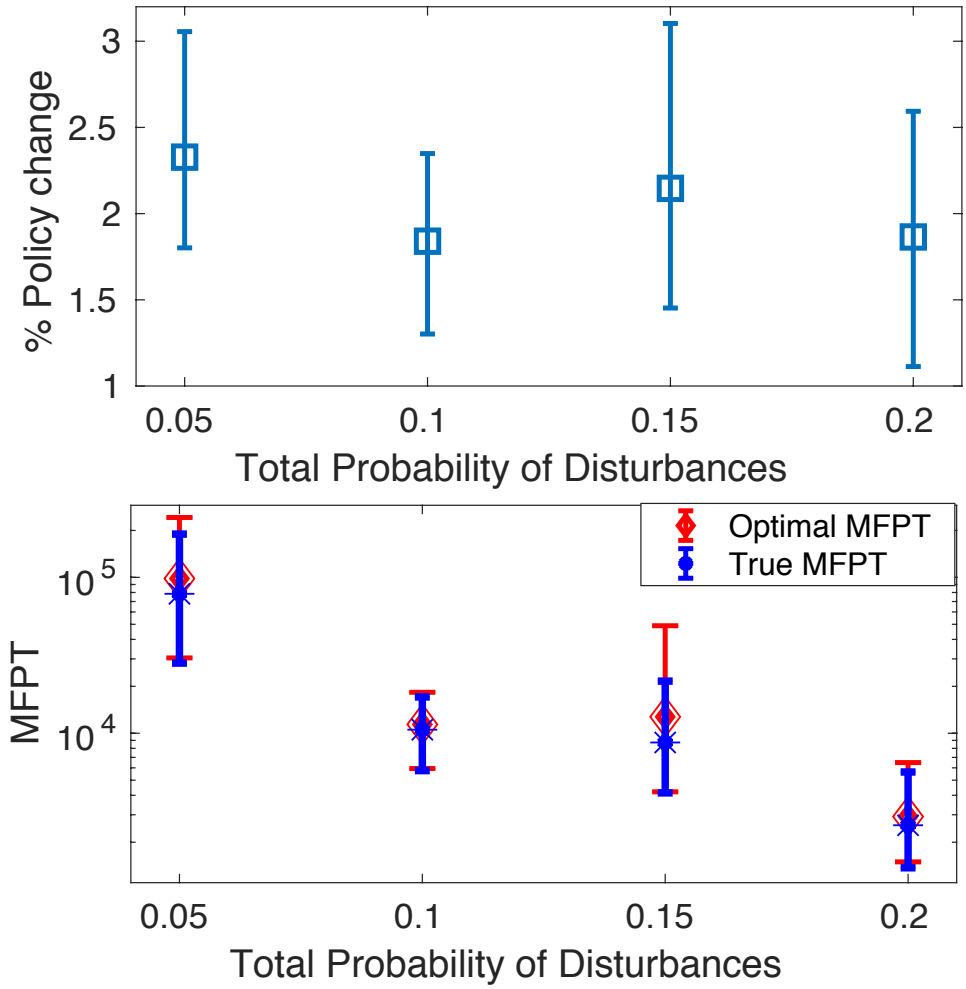

Figure 5.9: Sensitivity of optimal policy (left) and MFPT variability (right) to changes in probability of disturbances.

that the combined effects of magnitude as well as time of disturbance are significant to the performance of the system as the same magnitude disturbance accrues a different penalty if applied at different time. In addition we have also shown that the policy is quite robust to changes in probability distribution of the disturbances. In other words, even if the statistics of future disturbances are not well known, the policy derived from the wrong statistical assumptions will perform nearly as well as the true optimal policy.

In showing that our tools are applicable to random noise scenarios, we have cleared an important criterion in applying these tools to real world systems. So far we have

focused only on applying our tools to environmental disturbances. Future work will focus on adapting and improving these tools so that they can cope with changing or uncertain information about system parameters such as mass distribution and/or sensing capabilities as well as understanding the contracting nature of the closed-loop dynamics that allow us to apply our meshing based tools to analyze and improve the performance of high dimensional systems.

Chapter 6

Robustness Quantification of DRL based Policies for Biped Locomotion

In the last chapter we saw how meshing tools allows us to quantify robustness for high DOF biped robots. This is due to the fact that the low level controller forces the system dynamics to expand on a low dimensional manifold. In this chapter, we present a mesh-based approach to analyze stability and robustness of the policies obtained via deep reinforcement learning for various biped gaits of a five-link planar model. Intuitively, one would expect that including perturbations and/or other types of noise during training would likely result in more robustness of the resulting control policy. However, one would also like to have a quantitative and computationally-efficient means of evaluating the degree to which this might be so. Rather than relying on Monte Carlo simulations, which can become quite computationally burdensome in quantifying performance metrics, our goal is to provide more sophisticated tools to assess robustness properties of such policies. Our work is motivated by the twin hypotheses that contraction of dynamics, when achievable, can simplify the required complexity of a control policy and that control policies obtained via deep learning may therefore exhibit tendency to contract to lower-

dimensional manifolds within the full state space, as a result. The tractability of our mesh-based tools in this work provides some evidence that this may be so.

Model-based trajectory optimization methods, as described in [45, 17], involve the generation of a trajectory for the system that optimizes a certain cost function, such as energy minimization. This optimal solution is then used as a reference trajectory for the actual system through the use of a low level controller. The low level controller enforces contraction of the system onto a low dimensional manifold; e.g., for the underactuated dynamics, some subset of directly-controlled degrees of freedom (DOFs) converge rapidly to desired trajectories, compared to other (passive, slower, but still stable) DOF(s).

This contraction onto lower dimensional manifolds has quite some significance, as it can correspondingly allow for the avoidance of the classic "curse of dimensionality" in implementing discrete approximations to analyze the resulting nonlinear dynamics. In the present work, we hypothesize that such contraction may be likely, if not required, also to exist for policies that are (somewhat more mysteriously) derived from deep learning algorithms. Our reasoning is simply that it would be impossible to explore, let alone identify good control actions across, any non-trivial volume of a moderate or large dimensional state space. This implies that any seemingly well-behaved control policy for such systems displays a "contraction" behavior, in which the policy shepherds the dynamics to remain in lower-dimensional regions of state space. In this work, we identify and exploit such contraction, in turn applying meshing tools to evaluate the performance of control policies for such systems.

In the previous chapters, we have used model-based trajectory optimizations to corroborate human data studies with model-based energy-optimal gaits [46] and to explore non-intuitive motions for underactuated rolling locomotion systems [47]. While relationships between the cost function, constraints in the optimization, low level controller, and the robustness of the obtained policy often have some intuitive characteristics, it is not

so clear what quantitative effects the choice of reward function has on the robustness of the obtained policy. With this long-term goal in mind, and with application to quantification of deep learning policies as a particular aim, we analyze locomotion control polices obtained via deep reinforcement learning and propose the use of meshing tools to quantify stability and robustness in terms of failure rates.

6.1 5 link Biped Model in MuJoCo

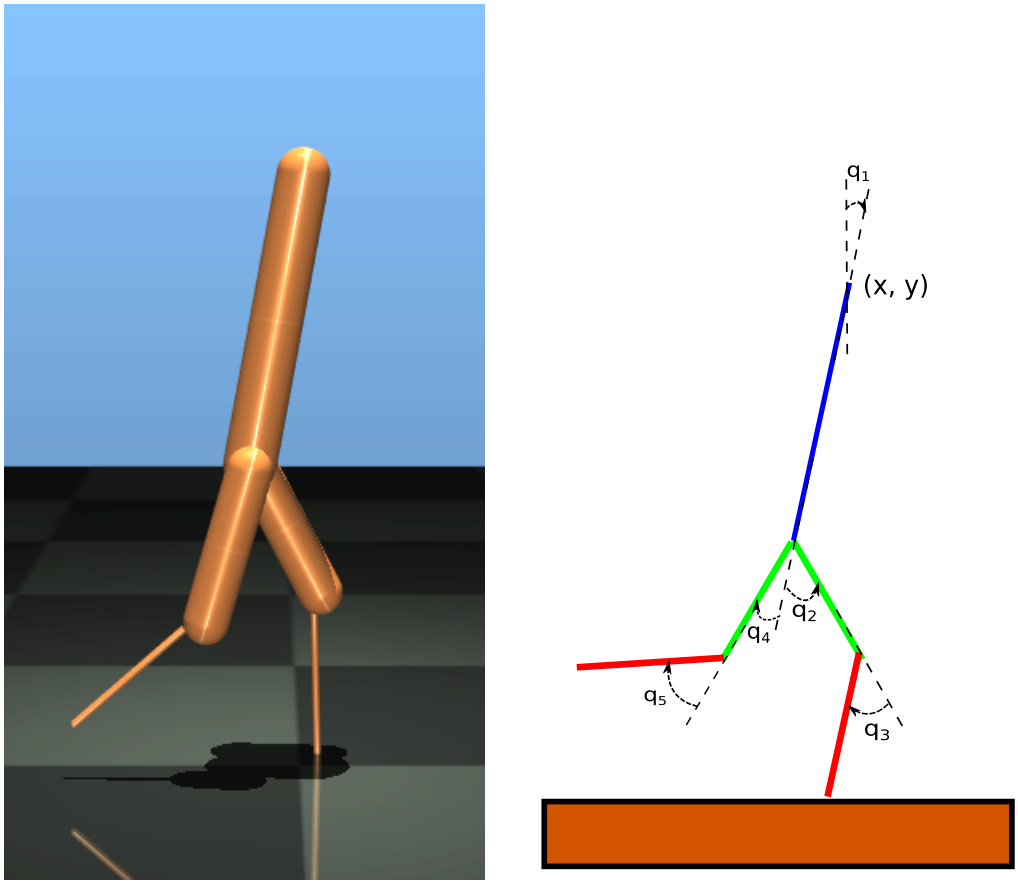

Figure 6.1: The 5-link biped model used in simulations. At left, the planar model in MuJoCo's simulation engine, used for our simulations. The stick figure on the right shows the degrees of freedom of the model on the left.

We use a 5-link planar biped model with point feet as shown in Fig. 6.1 for our

analyses. All simulations for the results presented in this chapter are done in MuJoCo [48]. The model has a torso, two hips and two knees. Its total mass is 70 (kg), and it stands 1.6 (m) tall when fully upright. Length, mass and inertia parameters, which are chosen to approximate a typical human, are identical to those listed for "Set 5" within [49]. The model has a total of 4 actuators (2 at hips and 2 at the knees) and 7 degrees of freedom $q := [x,\ y,\ q_1,\ q_2,\ q_3,\ q_4,\ q_5]^T$ where $[x,\ y]$ are the position coordinates of the topmost point of the torso and $[q_1,\ q_2,\ q_3,\ q_4,\ q_5]$ correspond to the five angles shown in the Fig. 4.1. We ignore rolling friction and set the friction model to oppose sliding in the tangential plane of two contacting bodies only. We set all the viscous damping coefficients to zero. All the contacts in the simulation are soft contacts: MuJoCo models the interaction between two bodies as a soft contact, for efficient computation. We set the integration method to Runge-Kutta (RK4) with a time step of 0.002 (s). All the torque inputs are restricted to $\pm$ 100 (N·m).

6.2 Control Policy

We use the Proximal Policy Optimization algorithm [23] in the openAI's baselines package [25] to train our model and obtain the control policies. The training algorithm models continuous time action space as a probability density distribution that it learns for each observation or state of the environment. While training, the algorithm samples from this distribution and then takes an action for the corresponding observation. During evaluation of a trained policy, however, instead of sampling from the learned distribution, we pick the action that has the maximum likelihood. Each control action is held at a constant value for a total of 4 consecutive time steps during simulation. Thus, even though the time step for the integration scheme we use is 0.002 (s), the action sampled for the current observation or the environment state is applied for a total of 0.008 (s). We

set the maximum training episode length to 4000 time steps, and each state observation is clipped to a maximum magnitude of 10.

6.3 Meshing

We create our mesh $\mathbb{M}$ as explained in Chapter 5 by deterministically mapping the reachable state space of the system on a Poincaré section for various disturbances, γ, which for this work are pushes of a variety of magnitudes and timing within the gait cycle. Because there are no constraints requiring left-right symmetry of the locomotion policy learned, we perform a Poincaré analysis on a full gait "cycle", i.e., after two steps are taken. A Poincaré section is taken when the left foot (an arbitrary choice) makes an impact with the ground. We denote the post-impact state as $s = x^+$. The mesh has one self-absorbing state (state #1) to which all failure events transition. In general, our meshing tools can allow us to calculate the optimal (switching) policy from a set of controllers, but for the current work, we simply analyze the policies individually obtained via DRL framework as explained in the section above. Once the left foot impacts the ground, we compare the post-impact state to previously observed post-impact states using the following metric:

$$d(s_i) = \min_{s_j \in \mathbb{M}} \sqrt{\sum_{k=1}^{n} (s_i(k) - s_j(k))^2}. \tag{6.1}$$

If $d(s_i) > d_{tr}$, where d_{tr} is some distance threshold, the state s_i is added to the mesh. A deterministic state transition matrix $T_{det}(s, \xi, \gamma)$ which describes all state transitions is maintained and updated at every iteration, where ξ is the particular DRL policy being analyzed. In case of failure, the corresponding transition goes to state #1, indicating that under the current control policy, the state has transitioned to the absorbing failure

state in the mesh.

Our disturbance set consists of various pushes γ happening at different times in forward as well as backward directions. Each push occurs at the center of mass (COM) of the torso link. The disturbance profile is characterized by a certain probability distribution that we choose, and it is denoted by $P(\gamma)$.

6.4 Experiments and Results

We perform analyses on two DRL-trained policies, obtained under different training conditions. The first policy is trained with a reward function that encourages forward velocity; there are no perturbations, and terrain is flat. At each time step, if the walker has not fallen, the reward is incremented by the forward velocity at that instant of time; the reward function also includes a penalty 1e-3 times the norm of the torques. For the second scenario, we use the same reward function, but we now introduce push disturbances while training (still on flat terrain). For both cases, we train several policies and then pick the one that has the maximum reward at the end of the training session for our analyses. Both cases also have the same coefficient of friction of 0.5 for contact between the ground and the walker model. For meshing, we ignore the x coordinate of the top of the torso, because we perform meshing for step-to-step transitions and it does not matter from what x position the model takes the step. The mesh thus contains 13 dimensional states, of which 12 are independent. (The left foot, but not necessarily the right one, must by definition be at $y = 0$ immediately post-impact, adding a constraint and removing an independent DOF on the Poincaré section.) We explore both cases and analyze the corresponding policies obtained in the next two subsections.

6.4.1 Case 1

In this scenario, we train the policy without any external perturbations, for flat ground walking. The policy with the best reward function, chosen for our analyses, results in a significantly asymmetric forward motion. As previously mentioned, we consider state-to-state transitions for a full gait cycle, defined as one complete sequence of right leg and subsequent left leg contacts with the ground. The states on the Poincaré section then represent the post impact state of the system each time the left leg makes contact with the ground. Fig. 6.3 illustrates a typical set of consecutive, post-impact states during locomotion (again, with no perturbations) for 250 gait cycles.

Figure 6.3 shows a non-uniform mesh of 64 points, achieved by applying our meshing algorithm to a set of 250 consecutive gait cycles, for the purposes of giving more intuition both into our meshing an into the lower-dimensional nature of these points. Here, sufficiently close neighbors within the full set of 250 states visited are "lumped together" at one of 64 total mesh points. Larger markers depict more frequently visited locations. For this particular figure, we performed a PCA analysis, to generate 3D axes to visualize the mesh. Here, three principle components account for 94% of the variance of the normalized Poincaré states. At top in Fig. 6.3, states fall near the depicted curvy, 2D surface within the 3D PCA space. Below this, straight lines show the step-to-step transitions, with terminal ends shown as thicker, blue segments. The gait has no observable limit cycle, yet it appears to be both stable (never-falling) – and chaotic.

As Figure 6.3 illustrates, applying the closed-loop DRL control policy to flat-ground walking with no noise inputs does not result in any discernable (period-n) limit cycle but instead exhibits chaotic behavior. Based on our numeric methods for mapping the reachable state space, post-impact states are bounded and seem to contract onto a lower-dimensional and bounded region of the full state space.

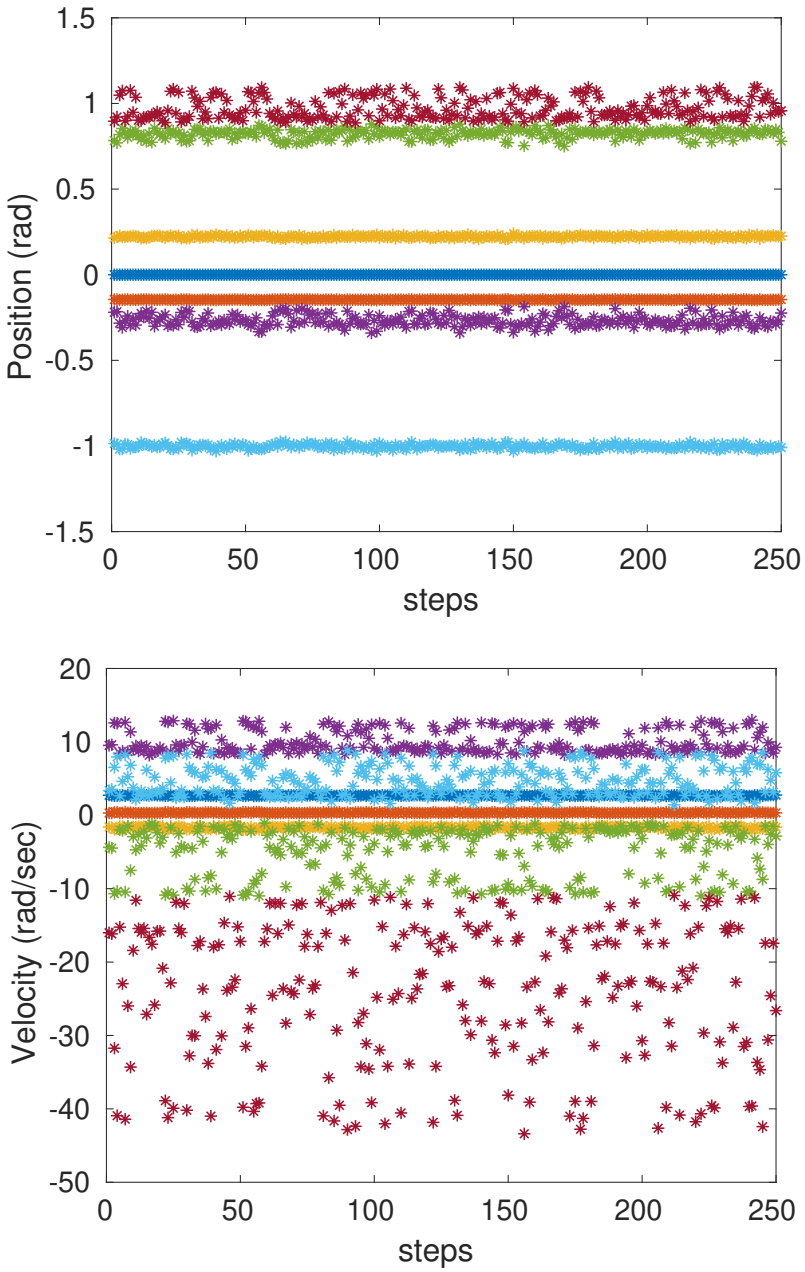

Figure 6.2: Plot of the post impact states on the Poincaré section for policy trained in case 1. The top figure plots the positions and the bottom figure plots the velocities.

To examine the robustness of the given policy to push disturbances, we proceed by calculating a mesh. From simulations, we find the complete two-step gait cycle takes about 0.5 (s). Based on this, for building the mesh, we define a threshold of 0.3 (s) after

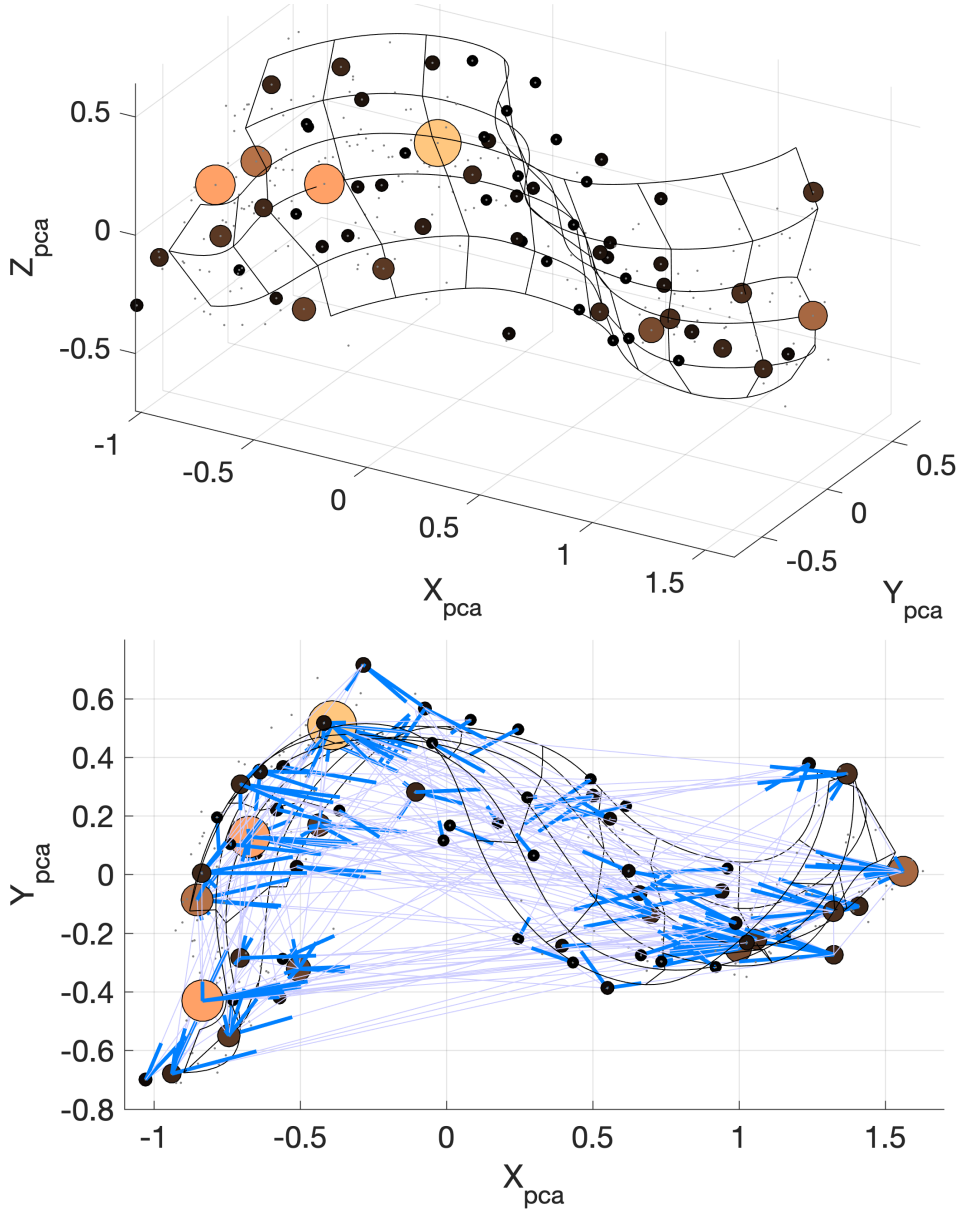

Figure 6.3: A principal component analysis (PCA) is used to visualize post-impact states visited for DRL-trained flat-ground locomotion, subject to no disturbances. See text for a detailed explanation.

which any post impact (for the left foot) state of the system, s_i, is added to the mesh if $d(s_i) > d_{tr}$, as described in the section above. Unless otherwise mentioned, all the meshing analyses presented here have been done for the threshold value of $d_{tr} = 0.6$. All push disturbances happen at the COM of the torso.

For illustrative purposes within Case 1, just four possible disturbance pushes are considered: a push of either +1000 (N) or −1000 (N) is applied at the COM of the torso for a duration of 0.008 (s), starting at either 0.1 or 0.25 (s) into the gait cycle.

The disturbance profile for the current mesh is as given in Table 6.1.

Disturbance	2	3	4	5
time (s)	0.1	0.25	0.1	0.25
magnitude (N)	1000	1000	-1000	-1000

Table 6.1: Disturbance Profile

Along with these 4 disturbances, we also consider a no-push disturbance, i.e., a push with 0 magnitude. The probability distribution for these disturbances is given by

$$P(\gamma) = \begin{cases} 0.4, & \text{if no disturbance} \\ 0.6/4, & \text{else.} \end{cases} \tag{6.2}$$

For the given policy and disturbance profile, we obtain a mesh with 28,757 states when $d_{tr} = 0.6$. The mean first passage time for the system for the given probability distribution comes out to only about 32 steps. Once the mesh is generated, our tools allow us to efficiently calculate the MFPT for any given probability distribution. For example, if we reduce the probability of a disturbance so that

$$P(\gamma) = \begin{cases} 0.8, & \text{if no disturbance} \\ 0.2/4, & \text{else,} \end{cases} \tag{6.3}$$

the MFPT increases to approximately 117 steps. This is intuitive as the probability of no push went up and since the total probability distribution has to sum to 1, the probability of the disturbance occurring went down. We also calculate a mesh for other values of d_{tr},

to analyze how varying this threshold distance changes the number of states in the mesh. If the mesh points occupy an n-dimensional subspace within the full state space, then the number of mesh points, N, required to span this subspace should grow as $N \propto d_{tr}^{-n}$, meaning a loglog plot of d_{tr} vs N would have slope $-n$. (See [50] for details.)

d_{tr}	0.6	0.7	0.8
N (# of mesh points)	28,757	14,891	8,517

Table 6.2: N versus d_{tr}, showing $n \approx 4.23$ for Case 1.

Table 6.2 shows this variation. A line fitting $x = log(d_{tr})$ vs $y = log(N)$ for these data has slope -4.2, showing that as we mesh more finely, the size of the mesh grows with dimensionality $n \approx 4.23$. Intuitively, the dimensionality of reachable state space will depend on both the sparsity of the perturbation set (recall we only include 4 non-zero perturbations here) as well as the contracting nature of the control policy. Rather than focusing on a more dense set of perturbations for Case 1, we instead focus on comparing the effects of adding perturbations during training (in Case 2). Note that although a coarse mesh is naturally less accurate than the fine mesh, it is still useful for getting a good idea of the trends that would occur with the finer mesh. A finer mesh is required for giving accurate guarantees on the performance, however, if only a general idea of the trend is needed then the trade off in the accuracy might be desirable. Currently the most time consuming part of our meshing tools is generating the mesh. Once the mesh is generated for the parameters of interest, all other steps in the analyses are significantly faster.

Figure 6.4 illustrates the distribution of mesh points for Case 1, using states with the greatest variance within the mesh ($\dot{q}_4$, $\dot{q}_5$, and $\dot{q}_1$) as three representative axes. Recall that even without added perturbations, the control policy results in a chaotic set of reachable states; these points fall within the blue region. The red points show states visited due to perturbations and from which failures are nearly certain. The step-to-step mixing effects

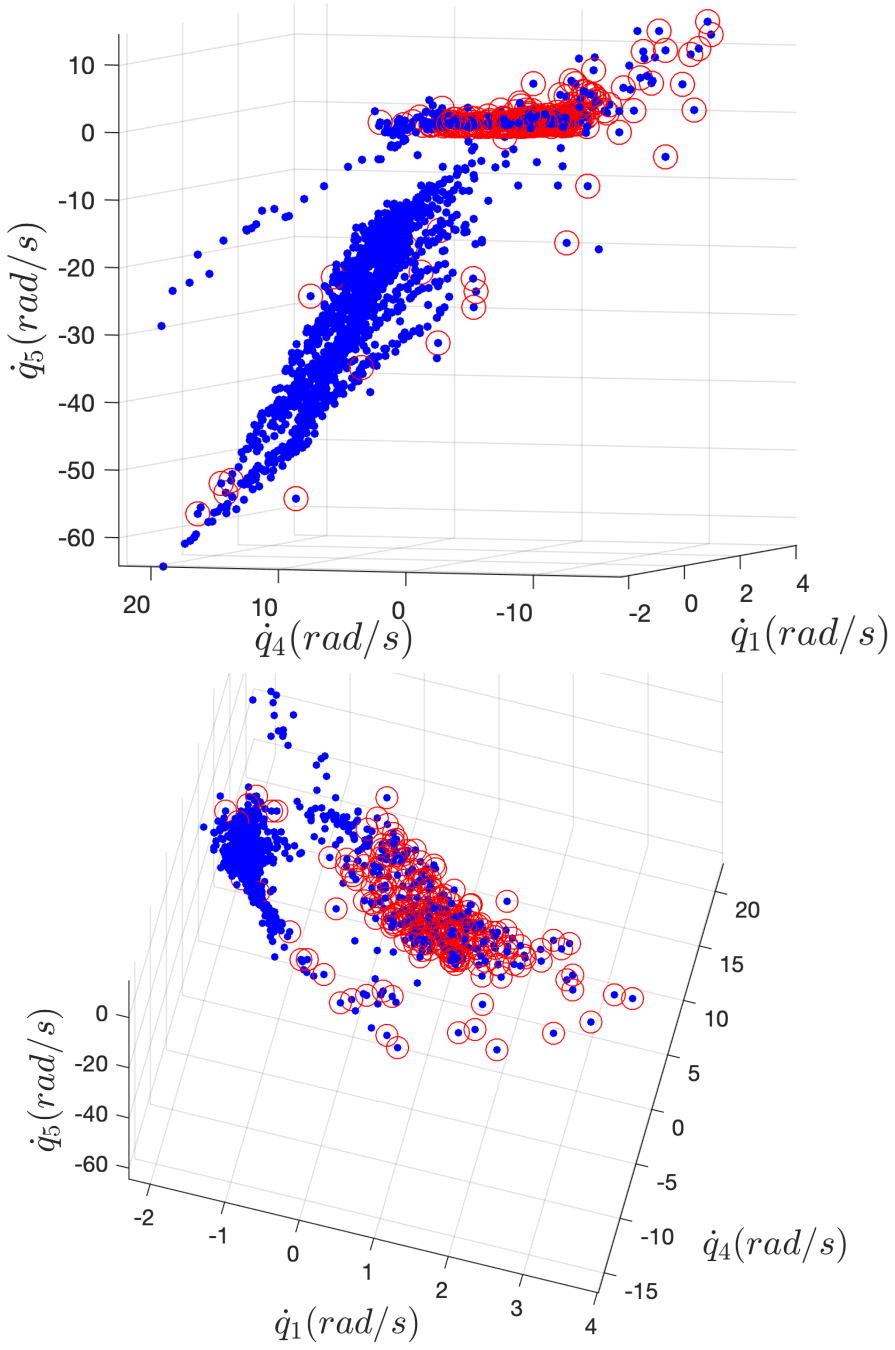

Figure 6.4: A 3D section of the full 13D Poincaré mesh states for Case 1. Subplots show two viewpoints of the same data. "Dangerous" states, with greater than 99% probability of failure on the next step, are circled in red.

of the perturbations push the system into the red regions shown, from which failures are nearly certain.

6.4.2 Case 2

In this scenario, we include push disturbances during the training, again using a reward function that primarily encourages forward motion, with a small penalty on energy use. Every 0.008 (s) interval during training, there is a 5 percent chance that the model will receive a push for the next 0.008 (s), with a 50/50 chance in either the forward or the back ward direction. We introduce pushes only at the COM of torso, and the magnitudes of the push are restricted to ± 1000 (N). As in Case 1, the policy with the maximum reward that we obtain for this new scenario also results in an asymmetric gait, now with a slower average gait cycle time of approximately 0.5 (s). We pick a threshold time of 0.45 (s) after which any post impact state (for left foot) with the ground is considered for meshing. To examine the effects of introducing disturbances during the training, we generate a mesh for the same disturbance profile (4 possible pushes, plus one no-push case) used in Case 1; for $d_{tr} = 0.6$, this leads to a mesh of 857 states. For the probability distribution given in Eq. 5.23, the MFPT for the system now comes out to about 10,467 steps, as compared with 32 steps for Case 1. An improvement in performance is an intuitively expected result, but our focus is on illustrating that our meshing tools allow us to quantify the improvement in robustness. We also generate the mesh for different d_{tr} values, shown in Table 6.3. All these meshes are also generated for disturbance profile shown in Table 6.1

d_{tr}	0.5	0.6	0.7
N (# of mesh points)	1705	857	574

Table 6.3: N versus d_{tr}, showing $n \approx 3.25$ for Case 2.

To analyze a more interesting disturbance profile and to check how the policy performs for magnitudes beyond the ones used for training, we generate a more complex mesh involving more varied disturbances, as shown in Fig. 6.5 As with the previous mesh, in

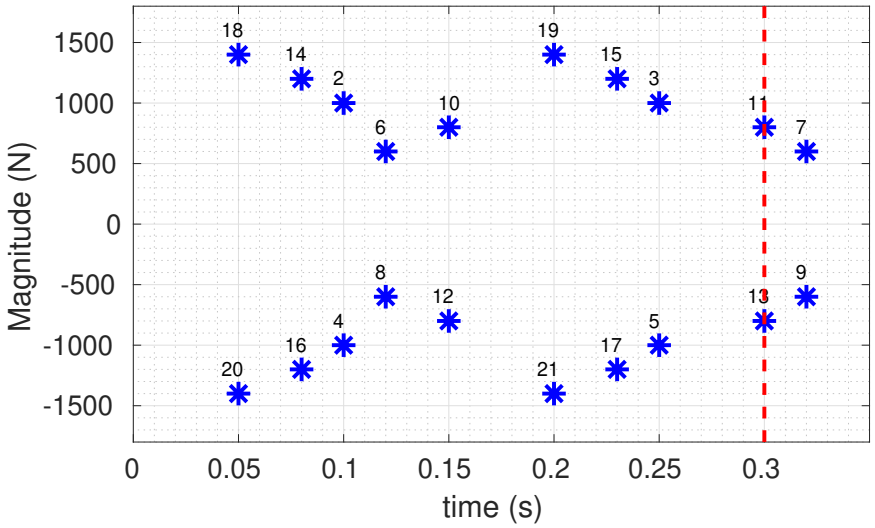

Figure 6.5: Disturbance profile for 20 push types. The dotted red line indicates the half gait cycle where the right leg makes contact with the ground

addition to the 20 disturbances shown, we also include a no push scenario. This mesh has a total of 20,660 states which is significantly higher than the previous mesh but because we explored the mesh for more disturbances, it also allows us to do much more interesting analyses. For example, if we set the probability distribution of the disturbances such that there is 0.6 probability (60% chance) of no push and 20% chance for disturbance 6 and disturbance 7 each, then we get a MFPT of infinity indicating that the walker will always recover under such disturbances. Similarly, if we set the probability disturbance such that probability of no push is 0.6, and probability of disturbance 8 and 9 is 0.2 each, then we get a MFPT of infinity as well. But, if we set the distribution such that probability of no push is 0.6 and the probability of disturbance 6, 7, 8 and 9 is 0.1 each, we get an MFPT of 32,260 showing that the *mixing effects* of disturbance 6, 7 and disturbance 8, 9 all combined over time will now create occasional failures, reducing the MFPT of the system significantly. Disturbances 6 and 7 correspond to pushes in the forward direction of magnitude 600 (N) and disturbance 8 and 9 correspond to the backward pushes of the same magnitude. We can also study the relative sensitivity of particular disturbances in

the profile. For this we take an example distribution given by:

$$P(\gamma) = \begin{cases} 0.4, & \text{no disturbance} \\ 0.5, & \text{disturbance of interest} \\ 0.1/19, & \text{else.} \end{cases} \tag{6.4}$$

A plot showing the corresponding performance of the DRL policy for Case 2 is shown in Fig. 6.6. This is an interesting plot because it shows the coupled effect of direction and time of impact of the disturbance matters significantly. For example, we see that some disturbances of higher magnitude have a better MFPT than some disturbances of lower magnitude because they happen at different time instances.

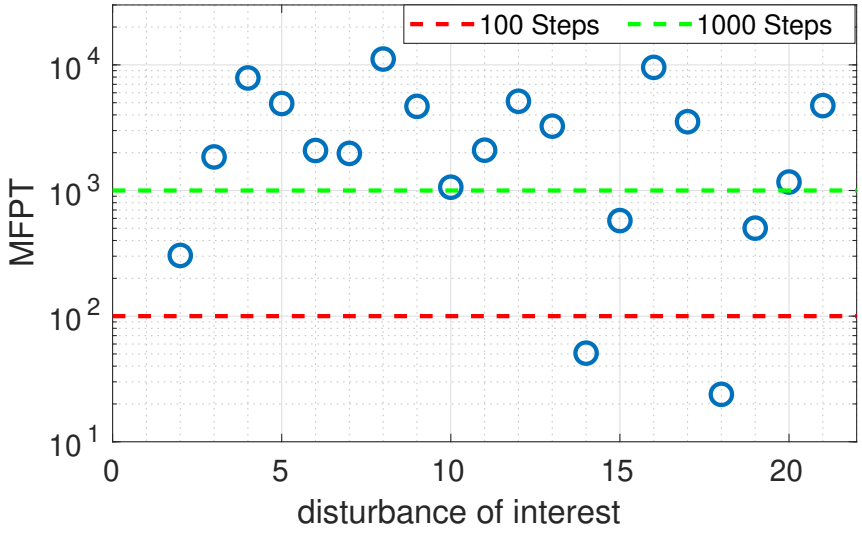

Figure 6.6: MFPT variations as one disturbance of interest becomes more likely. The MFPT is shown on log scale to make the plot more readable.

Figures 6.7 and 6.8 show subsets of the full mesh corresponding to the reachable state space when each of two different subsets of the disturbance profiles shown in Fig. 5.2 are allowed. For Fig. 6.7, we exclude any pushes occurring before 0.1 seconds in the gait cycle; i.e., profiles 14, 16, 18 and 20 are excluded. For Fig. 6.8, all 20 push disturbances

are included. For both figures, we assume a 2% chance of a push during each gait cycle, with the push type drawn with uniform probability from the allowable subset of pushes.

In Fig. 6.7, both subplots show the mesh points visited for this noise case, with darker points representing much more frequently-visited states. At right, magenta "+" symbols are overlaid to show a set of 250 consecutive states visited (chaotically) when there is no noise during post-training testing of the policy. The MFPT predicted by the mesh is about 958,000 gait cycles. Once all 20 pushes are allowed (Fig. 6.8), this drops to a MFPT of only about 4,900 gait cycles. In this latter case, we can see that the system now visits a significant number of "unsafe" states (shown in red) that department significantly from the chaotic variability of locomotion when there is no noise.

6.5 Dimensionality analysis

One important result we would like to analyze is the correlation between robustness and dimensionality. The model that we analyze in this chapter has a 14 dimensional state space. We do our meshing analysis by completely exploring the reachable state space of the system. Ideally, it should be infeasible to explore the reachable state space of a 14 dimensional system. But as we have seen in the previous chapter, the low level controller forces the states to expand on a much lower dimensional manifold than the entire state space. This makes it possible for us to apply our meshing approach to higher dimensional systems. Even for deep reinforcement learning control policies where we do not see an obvious limit cycle behavior on the Poincaré section, the policies still force the system dynamics to expand on a lower dimensional manifold. Fig.6.9 shows the dimensionality for policies trained with and without noise. We see that the dimensionality for policy trained without noise is 4.23 and for the policy trained with noise is 3.25 which is significantly less than 14. This is significant but another important thing to note

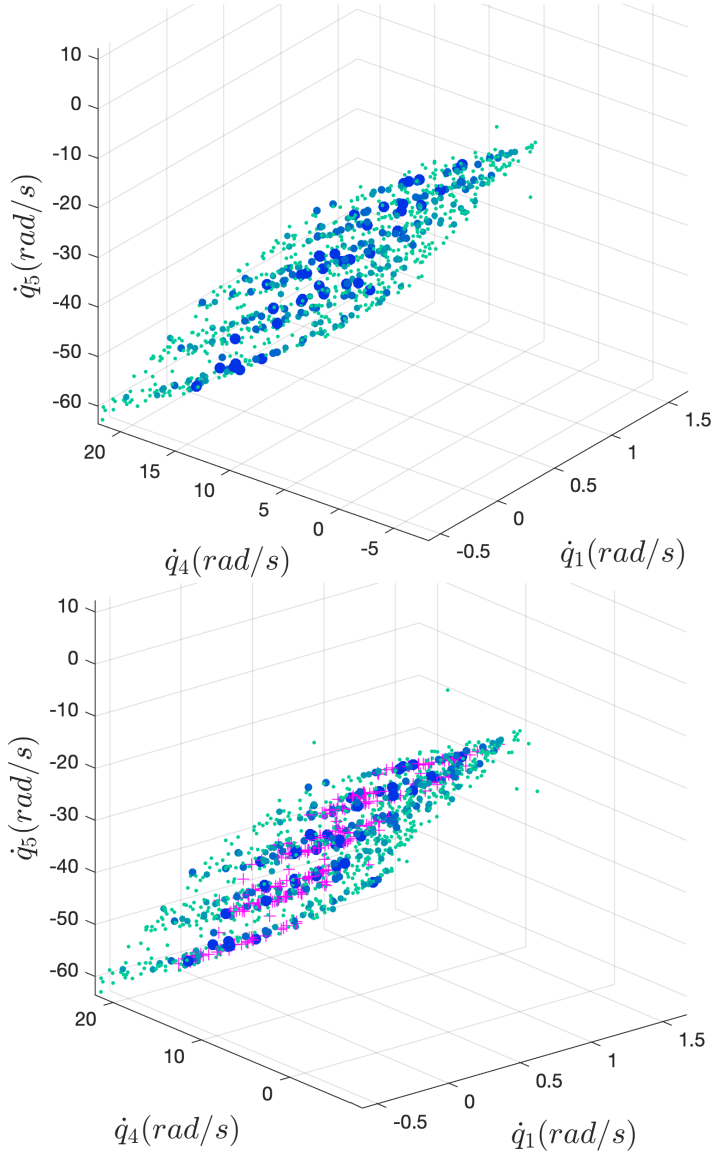

Figure 6.7: A 3D slice of the full 13D Poincaré states generated to mesh the policy trained in Case 2, when all but disturbance profiles 14, 16, 18 and 20 are included, i.e., excluding pushes occurring before 0.1 seconds.

here is the reduced dimensionality of the policy trained with disturbance because we know that this policy is much more robust to disturbances. We saw similar trend in the last chapter where the dimensionality for double support phase trajectories is slightly lesser than that for the single support phase. This suggests that there is indeed a strong correlation between the dimensionality and robustness of the system. More experiments

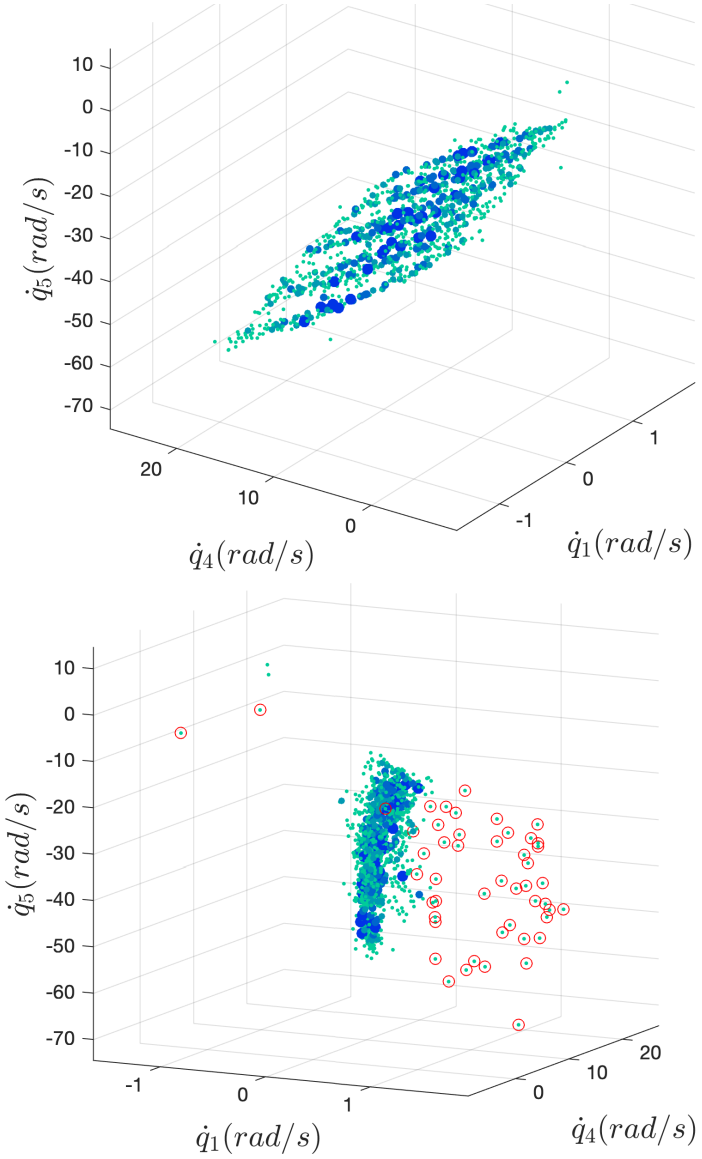

Figure 6.8: A 3D slice of the full 13D Poincaré states generated to mesh the policy trained in Case 2, when all 20 timing and magnitude combinations shown in Fig. 5.2 are included. As in Fig. 6.4, the subplot at right highlights states from which immediate failure has probability greater that 99%.

are definitely needed to analyze the exact nature of this relationship and the existence of such a correlation can be useful in further quantifying and improving the performance of the system

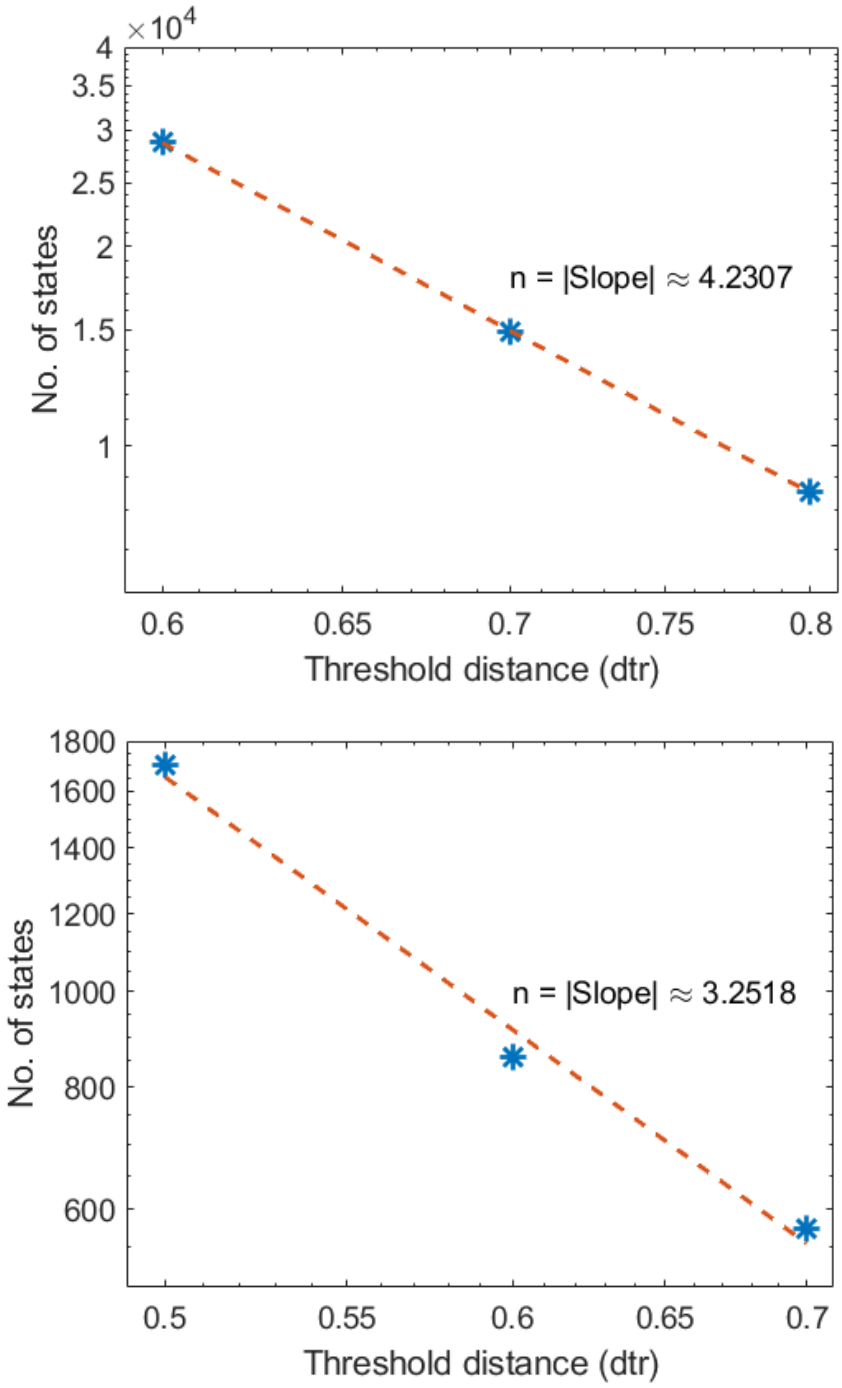

Figure 6.9: Dimensionality of policies trained with deep reinforcement learning with no noise (top) and with noise (bottom)

6.6 Conclusion

We have demonstrated the effectiveness of our meshing tools in analyses of the polices created via deep reinforcement learning. To quantify the robustness of these policies, we create meshes for different disturbance combinations that can happen at various times in the gait cycle and then set transitions among mesh points based on particular assumed probability distributions of the disturbances. The policy trained for only forward motion is significantly less robust to perturbations as compared to the policy trained with impacts, which is intuitive but our tools gave us an estimate on the actual performance improvement. Such an estimate is important if we want to improve the performance of training by changing various parameters that can affect the outcome of the trained policy. In addition, by performing the analyses on various probability distributions we show that our tools also allow us to plot performance trends which can help us to understand the effect each individual disturbance in the disturbance profile has on the overall performance of the policy. We also show how the mixing effects of these disturbances can have significant effect on robustness.

In showing the applicability of our meshing tools to analyze the policies obtained via deep reinforcement learning, we have also created an indirect feedback loop that can be used to improve the performance of the policies by tuning various elements of the training framework. A future goal is to integrate this feedback loop within the training framework. We also plan to explore how increasing the number of discrete perturbation types affects mesh size. Although we consider toy sets of perturbations here, for illustrative purposes, practical use of these tools should cope with denser sets of perturbations. Finally, another important goal is to explore how various reward functions and/or disturbances during training may increase the degree to which a DRL policy contracts the dynamics; reducing the dimensionality of the state space visited would

increase the practicality of meshing. We show some initial success in this by reducing dimensionality from $n = 4.23$ to $n = 3.25$ when training includes pushes. Future work is planned to investigate if this a repeatable trend and whether more extreme perturbations during training can further improve contraction.

Chapter 7

Conclusion and Future Work

Legged locomotion has been a long standing area of research in robotics. While significant strides have been made in humanoid locomotion over the past years, there are still a lot of challenges before legged robots become viable in everyday life. In this thesis we try to overcome some of these challenges by using optimization and deep reinforcement learning for generating motion for a 5 link planar biped model and using meshing tools to quantify the robustness of these policies. We also study the effects of system parameters and control policies on the energy efficiency on robustness of the system.

In Chapter 3 our simulations show an increased impact of adding mass distally, near the ankle and foot. Interestingly our data also show a surprisingly low burden associated with a location just above the knee. These findings deserve further study and may significantly influence design of future exoskeletons, particularly when power must be carried on board. Our studies have been designed to expand and refine the augmentation factor equation such that future revisions to the equation building upon this work should enable designers to quantitatively balance the power and mass of an exoskeleton more effectively. This understanding will allow exoskeleton designers to optimize performance more effectively, minimizing the arduous prototype and test cycle and making the design

and construction of exoskeletons more efficient.

In Chapter 4 we demonstrate that although optimizing for energy alone and then stabilizing the trajectories can work, there is a need for a more cohesive framework that takes into account both energy as well as stability of the system for optimization simultaneously. We also see that different physical parameters have varying effects on stability and energy and more comprehensive frameworks that take this into account are needed. To understand the underlying nature of tradeoff between all these parameters, we present simulation data for 5 different sets of parameters. From our experiments, we found that the mass distribution closest to that of a human being (set 5) is the most energy efficient but not the most robust. The most robust is mass set 2. These phenomena may in part be a result of other factors, such as choice of feedback control structure and certainly warrant further study. To explore trade-offs, we presented COT vs rate of convergence in Figure 4.12, which illustrates a Pareto frontier, formed essentially by sets 2, 3, and 5, while sets 1 and 4 provide poor trade-off characteristics by comparison.

In chapter 5 we use our meshing tools for improving and analyzing the performance for walking on rough terrain to improve the performance of the biped walker in the presence of more random real world noise scenarios like push disturbances. To demonstrate the effectiveness of our tools, we perform experiments on single support phase and double support phase trajectories. Intuitively, DS trajectories woulbe be more robust than SS trajectories in the presence of disturbance, but our tools allow us to quantify this performance gain which is very important as a metric. Our tools also allow us to do sensitivity analyses for different kinds of disturbances. Our plots show that disturbances with similar magnitude have different effects on the system when applied at different time intervals indicating that the combined effects of these factors are significant. Our sensitivity analyses also show that our policy is quite robust to changes in the probability distribution of the disturbance. Our experiments show that we can effectively use

our meshing tools to analyze robustness of biped systems in the presence of real world disturbances.

In Chapter 6, we further demonstrate the effectiveness of our meshing tools in analyses of the polices created via deep reinforcement learning. To analyze the robustness, we build a mesh from the states on the Poincaré section in the presence of several push disturbances. Our experiments show that the policy trained without disturbance is not as robust compared to the one trained with disturbances. This is expected but the important point is that our tools allow us to quantify this improvement. Such quantification is important if we want to build better control techniques and tools. Our tools also allow us to study individual effects of a specific disturbance on the performance of the system as well as the combined effect effect of a group of disturbances. This is because once the mesh is completely constructed, we can change the probability of disturbances to analyze particular trends. Our experiments demonstrate the versatility and effectiveness of our tools in analysing deep reinforcement learning control policies and provide an insight into how such policies can be analyzed.

We have studied the effect of system parameters on energy and stability of the biped system. One of our future goal is to build a more comprehensive framework that will take into account all these parameters for motion planning. An important step in this direction was demonstrating the effectiveness of our meshing tools. In showing the applicability of our meshing tools to analyze the control policies in the presence of random real world disturbances, we have also created an indirect feedback loop that can be used to improve the performance of the policies by tuning various elements of the control framework. Our future goal is to integrate this feedback loop within the optimization or the training framework. We also plan to explore how increasing the number of discrete perturbation types affects mesh size. Although we consider toy sets of perturbations in this thesis, for illustrative purposes, practical use of these tools should cope with denser sets of

perturbations.

So far we have only focused on applying our tools to environmental disturbances. Future work will also focus on adapting and improving these tools so that they can cope with changing or uncertain information about system parameters such as mass distribution and/or sensing capabilities as well as understanding the contracting nature of the closed loop dynamics that allow us to apply our meshing based tools to analyze and improve the performance of high dimensional systems.

Finally, another important goal is to explore how various reward functions or cost functions and/or disturbances during training and optimization may increase the degree to which a control policy contracts the dynamics; reducing the dimensionality of the state space visited would increase the practicality of meshing. We show some initial success in this in our DRL policies by reducing dimensionality from $n = 4.23$ to $n = 3.25$ when training includes pushes. Future work is planned to investigate if this a repeatable trend and how strongly correlated is the connection between contraction onto a lower dimensional manifold and stability.

Appendix A

Biped Model

The methods described in this thesis are applicable to a wide variety of systems but in order to demonstrate their effectiveness we perform experiments on the 5 link planar biped model as shown in Fig. 4.1. As the model is planar, locomotion is constrained to the sagittal plane. Our model largely remains same across the experiments that we perform with certain modifications. In Chapter 3 we use a 7 link model which was obtained by adding curved feet to the 5 link model. It is important to note that even though the feet are curved, the contact is a point and the model is underactuated. In Chapter 5 we use the 5 link model with 7-DOF as we allow for more complicated motions such as sliding and slipping. This increases the dimensionality of the state space from 10 to 14. In Chapter 6 we use a 5 link model in MuJoCo which is similar to our 5 link model created in matlab. The mathematical model is given by 5.1. All our experiments are performed on flat terrain and we use an inelastic collision model to calculate the post impact states for the biped after the swing leg impacts the ground. Specifically, we use

Eq. A.1 to calculate the post impact states.

$$\begin{bmatrix} D(q^-) & -E_2(q^-)' \\ E_2(q^-)' & 0_{2\times 2} \end{bmatrix} \begin{bmatrix} \dot{q}^+ \\ F_2 \end{bmatrix} = \begin{bmatrix} D(q^-)\dot{q}^+ \\ 0_{2\times 1} \end{bmatrix} \tag{A.1}$$

Here, $E_2(q) = \frac{\partial}{\partial q} p_2(q)$ where $p_2(q)$ is the tip of the swing foot with respect to the inertial frame of reference. F_2 is the force applied at the tip of the swing foot. This process is explained in detail in [43].

Paramater	Label	Value
Torso mass	M_T	50 (kg)
Femur mass	M_f	7 (kg)
Tibia mass	M_t	3 (kg)
Torso length	l_T	0.77 (m)
Femur length	l_f	0.4 (m)
Tibia length	l_t	0.43 (m)
Torso inertia	I_T	2.595 (kg.m^2)
Femur inertia	I_f	0.097 (kg.m^2)
Tibia inertia	I_t	0.0481 (kg.m^2)
Torso radius	r_T	0.1 (m)
Femur radius	r_f	0.05 (m)
Tibia radius	r_t	0.05 (m)
Acceleration due to gravity	g	9.8 (m/s^2)

Table A.1: Length and COM parameters used in simulation experiments

Unless otherwise mentioned, table A.1 lists the parameters for the model we use in our experiments. We calculate the moment of inertia for the links by assuming the geometry

of a cylinder. For our actuators, we have 2 motors at the hips and 2 at the knees. For our experiments, we ignore the damping effects at the joints. For friction, we assume a coefficient of friction μ to be 0.5. We only assume sliding friction and ignore the rolling friction in our simulations.

Appendix B

Direct transcription Methods

We have discussed our optimization approach in detail in Chapter 2. Here, we provide more specific details regarding the implementation of the actual techniques. As mentioned in Chpater 2 we use CasADi [31] to implement our direct transcription optimization techniques as it allows us to calculate accurate gradients very efficiently using algorithmic differentiation. CasADi allows for a variety of optimization solvers to be integrated with it. Our choice of solver for our experiments was IPOPT [33] as it gave us the best rate of convergence and repeatability for the solution. CasADi is a library that can be integrated with matlab, python, C++ etc. We use the matlab library as all our simulations and analyses are performed in matlab. IPOPT calculates the required hessian for the problem using a recursive BFGS update.

Table B.1 shows the lower and upper bounds on the variables used in optimization experiments performed in Chapter 4. The open variables for the optimization in this case are the 10 state variables and the 4 input variables. These bounds are imposed on the variables at each knot point on the trajectory. The total time of the trajectory is 0.6 (s) and we chose the integration time step to be 0.01 (s) which gives us 60 knot points along the entire trajectory. Table B.2 lists the size of the optimization problem in terms of

Paramater	Label	Lower Bound	Upper Bound
Torso angle	q_5	-0.78 (rad)	0.78 (rad)
Stance Femur angle	q_1	3.14 (rad)	4.71 (rad)
Stance Tibia angle	q_3	-0.35 (rad)	0 (rad)
Swing Femur angle	q_2	1.57 (rad)	3.14 (rad)
Swing Tibia angle	q_4	-0.35 (rad)	0 (rad)
Torso angular velocity	$\dot{q_5}$	-12.56 (rad/s^2)	12.56 (rad/s^2)
Stance Femur angular velocity	$\dot{q_1}$	-12.56 (rad/s^2)	12.56 (rad/s^2)
Stance Tibia angular velocity	$\dot{q_3}$	-12.56 (rad/s^2)	12.56 (rad/s^2)
Swing Femur angular velocity	$\dot{q_2}$	-12.56 (rad/s^2)	12.56 (rad/s^2)
Swing Tibia angular velocity	$\dot{q_4}$	-12.56 (rad/s^2)	12.56 (rad/s^2)
Input Torques	$u_1 - u_4$	-100 (N.m)	100 (N.m)

Table B.1: Lower and Upper bounds on variables for optimization performed in Chapter 4 for a stride length of 0.6m

various statistics. We see that the total number of variables is 855 and the total equality constraints are 613 which are less than the variables. This is a required condition for the optimization problem to be well defined.

The main difference between the optimization in Chapter 4 and Chapter 5 is that the state dimension increases from 10 to 14 as we allow for more complicated motion such as sliding and slipping. This increases the size of our optimization problem. Moreover, because we allow for slipping and sliding, we also leave the forces at the contact point as open variables. This is because it is easier to put bounds on the variables than add them as constraints. To make the motion physically consistent, we add constraints on the forces such that at the point of contact the force in the y direction is always positive

Paramater	Value
Total No. of variables	855
Variables with only lower bound	0
Variables with only upper bound	0
Variables with upper and lower bound	854
Equality constraints	613
Inequality constraints	366
Inequality constraints with only lower bound	306
Inequality constraints with only upper bound	0
Inequality constraints with lower and upper bound	60
Nonzero elements in equality constraint Jacobian	8477
Nonzero elements in inequality constraint Jacobian	1943
Nonzero elements in Lagrangian Hessian	6777

Table B.2: Problem statistics for optimization experiments in Chapter 4

and the force in the x direction is such that it satisfies the no slip condition. The no slip condition is enforced in our optimization using the Eq. B.1

$$\begin{aligned} \mu F_y + F_x > 0 \\ \mu F_y - F_x > 0 \end{aligned} \tag{B.1}$$

Also, it is important to note that, when imposing the conditions given by Eq. B.1, we chose the value of μ to be much smaller than the actual value. This is to account for deviation that occurs while implementing the optimal trajectory on the system using a low level controller. Due to these additional variables and constraints, the size of our optimization problem increases as is evident from Tables B.3 and B.4. Our total number

of variables is now 1245 which is much more than 855. This added complexity increases the computation time required but it has little effect on the convergence of the solver.

Paramater	Label	Lower Bound	Upper Bound
Torso angle	q_5	-0.3 (rad)	0.3 (rad)
Stance Femur angle	q_1	1.05 (rad)	5.76 (rad)
Stance Tibia angle	q_3	-0.78 (rad)	0.05 (rad)
Swing Femur angle	q_2	1.05 (rad)	5.76 (rad)
Swing Tibia angle	q_4	-0.35 (rad)	0.05 (rad)
Stance foot tip x-location	x_e	0 (m)	0 (m)
Stance foot tip y-location	y_e	0 (m)	0 (m)
Torso angular velocity	$\dot{q_5}$	-18.85 (rad/s^2)	18.85 (rad/s^2)
Stance Femur angular velocity	$\dot{q_1}$	-18.85 (rad/s^2)	18.85 (rad/s^2)
Stance Tibia angular velocity	$\dot{q_3}$	-18.85 (rad/s^2)	18.85 (rad/s^2)
Swing Femur angular velocity	$\dot{q_2}$	-18.85 (rad/s^2)	18.85 (rad/s^2)
Swing Tibia angular velocity	$\dot{q_4}$	-18.85 (rad/s^2)	18.85 (rad/s^2)
Stance foot tip x-velocity	x_e	0 (m/^{2})	0 (m/^{2})
Stance foot tip y-velocity	y_e	0 (m/^{2})	0 (m/^{2})
Force along x-axis at the contact	F_x	-∞ (N)	∞ (N)
Force along the y-axis at the contact	F_y	200 (N)	5000 (N)
Input Torques	$u_1 - u_4$	-200 (N.m)	200 (N.m)

Table B.3: Lower and Upper bounds on variables for optimization performed in Chapter 5 for a stride length of 0.6m

Additionally, even though we allow for the model to slip and slide, for the purpose of optimization, we generate the motion using no slip condition at the stance leg. This is

Paramater	Value
Total No. of variables	1245
Variables with only lower bound	0
Variables with only upper bound	0
Variables with upper and lower bound	927
Equality constraints	1107
Inequality constraints	473
Inequality constraints with only lower bound	393
Inequality constraints with only upper bound	0
Inequality constraints with lower and upper bound	80
Nonzero elements in equality constraint Jacobian	14393
Nonzero elements in inequality constraint Jacobian	1943
Nonzero elements in Lagrangian Hessian	9666

Table B.4: Problem statistics for optimization experiments in Chapter 5

evident by the bounds shown in Table B.3 where we bound the x and y velocity of the tip of the stance foot to 0.

Appendix C

Deep Reinforcement Learning for 5 Link Planar Biped Model

For our experiments in this thesis, we use the openai baselines package [25] to train the 5 link planar biped model for locomotion on flat terrain. The model constructed for the training is created in a physics simulator called MuJoCo [48] which has become one of the most popular platforms to use for deep reinforcement learning due to its speed. Specifically we use the Proximal Policy Optimization (PPO2) [23] algorithm to generate the policies. The policy takes in observations from the environment and generates actions based on these that try to maximize the reward that the policy has been trained for. The reward in this case is the forward velocity i.e the policy has been trained to make the model locomote as fast as possible in the x-direction. The observation for our policy is a 13 dimensional state vector of the system where we ignore the x position of the model as the current x position of the model should have no effect on the actions generated if the policy is trying to maximize the speed in the x-direction. The time step for our simulation model is 0.002 (s). For the purposes of training, we hold the actions generated for a particular observation for a total of 4 time steps. So during training, any action

that is generated by the policy is applied to the model for 0.008 (s) after which a new action set is generated based on the observation at that instant of time. Table C.1 gives the hyperparameters used in the training.

Hyperparameter	Value
Horizon	2048
Num. Environments	6
Adam step size	3×10^{-4}
Num. epochs	4
Minibatch Size	3072
Discount Factor (γ)	0.99
GAE parameter (λ)	0.95

Table C.1: Hyperparameters used for training of the 5 link planar Walker in MuJoCo.

Bibliography

[1] Wikimedia Commons, *File:asimo 4.28.11.jpg*, 2011. [Online] https://commons.wikimedia.org/wiki/File:ASIMO_4.28.11.jpg.

[2] Wikimedia Commons, *File:9 pic.png*, 2018. [Online] https://commons.wikimedia.org/wiki/File:9_pic.png.

[3] Wikimedia Commons, *File:fractaldimensionexample.png*, 2017. [Online] https://commons.wikimedia.org/w/index.php?title=File:Fractaldimensionexample.PNG&oldid=247697185.

[4] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, *Efficient bipedal robots based on passive-dynamic walkers*, *Science* **307** (2005), no. 5712 1082–1085.

[5] O. Von Stryk and R. Bulirsch, *Direct and indirect methods for trajectory optimization*, *Annals of Operations Research* **37** (1992), no. 1 357–373.

[6] M. Hardt, K. Kreutz-Delgado, and J. W. Helton, *Optimal biped walking with a complete dynamical model*, in *Proc. IEEE Conference on Decision and Control (CDC)*, vol. 3, pp. 2999–3004, 1999.

[7] C. Paul and J. C. Bongard, *The road less travelled: Morphology in the optimization of biped robot locomotion*, in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, pp. 226–232, 2001.

[8] E. Westervelt and J. Grizzle, *Design of asymptotically stable walking for a 5-link planar biped walker via optimization*, in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 3117–3122, 2002.

[9] D. Djoudi, C. Chevallereau, and Y. Aoustin, *Optimal reference motions for walking of a biped robot*, in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 2002–2007, 2005.

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

[11] S. Kuindersma, F. Permenter, and R. Tedrake, *An efficiently solvable quadratic program for stabilizing dynamic locomotion*, in *Proc. IEEE Int. Conf. on Robotics and Auto. (ICRA)*, pp. 2589–2594, IEEE, 2014.

[12] A. Majumdar and R. Tedrake, *Funnel libraries for real-time robust feedback motion planning*, *The International Journal of Robotics Research* **36** (2017), no. 8 947–982.

[13] Q. Nguyen, A. Agrawal, X. Da, W. C. Martin, H. Geyer, J. W. Grizzle, and K. Sreenath, *Dynamic walking on randomly-varying discrete terrain with one-step preview*, in *Robotics: Science and Systems*, 2017.

[14] K. Byl and R. Tedrake, *Metastable walking machines*, *I. J. Robotics Res.* **28** (2009) 1040–1064.

[15] C. O. Saglam and K. Byl, *Quantifying the trade-offs between stability versus energy use for underactuated biped walking*, in *Proc. IEEE/RSJ Int. Conf. on Intell. Robots and Systems (IROS)*, pp. 2550–2557, 2014.

[16] K. A. Hamed, B. G. Buss, and J. W. Grizzle, *Exponentially stabilizing continuous-time controllers for periodic orbits of hybrid systems: Application to bipedal locomotion with ground height variations*, *The Int. J. of Robotics Research (IJRR)* **35** (2016), no. 8 977–999.

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

[18] W. Xi, Y. Yesilevskiy, and C. D. Remy, *Selecting gaits for economical locomotion of legged robots*, *The International Journal of Robotics Research (IJRR)* **35** (2016), no. 9 1140–1154.

[19] S. Sovero, N. Talele, C. Smith, N. Cox, T. Swift, and K. Byl, *Initial data and theory for a high specific-power ankle exoskeleton device*, in *Proc. Int. Symp. on Experimental Robotics (ISER)*, pp. 355–364, 2016.

[20] G. Bellegarda, N. Talele, and K. Byl, *Exploring nonintuitive optima for dynamic locomotion*, 2017 (submitted).

[21] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, *Emergence of locomotion behaviours in rich environments*, 2017.

[22] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, *Trust region policy optimization*, 2015.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017.

[24] J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess, *Learning human behaviors from motion capture by adversarial imitation*, 2017.

[25] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines." `https://github.com/openai/baselines`, 2017.

[26] X. Da and J. Grizzle, *Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots*, 2017.

[27] G. Bellegarda and K. Byl, *Combining benefits from trajectory optimization and deep reinforcement learning*, 2019.

[28] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, second ed., 2010.

[29] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[30] M. Srinivasan, *Why walk and run: energetic costs and energetic optimality in simple mechanics-based models of a bipedal animal*. Cornell University Ithaca, NY, 2006.

[31] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, *CasADi – A software framework for nonlinear optimization and optimal control*, *Math. Programming Computation* (In Press, 2018).

[32] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, *Mathematical programming* **106** (2006), no. 1 25–57.

[33] L. T. Biegler and V. M. Zavala, *Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization*, *Computers & Chemical Engineering* **33** (2009), no. 3 575–582.

[34] H. Mittleman, *Decision tree for optimization software*, 2017. (Benchmarking for SQP, and other optimization problems).

[35] M. Wehner, B. Quinlivan, P. M. Aubin, E. Martinez-Villalpando, M. Baumann, L. Stirling, K. Holt, R. Wood, and C. Walsh, *A lightweight soft exosuit for gait assistance*, in *2013 IEEE international conference on robotics and automation*, pp. 3362–3369, IEEE, 2013.

[36] L. M. Mooney, E. J. Rouse, and H. M. Herr, *Autonomous exoskeleton reduces metabolic cost of human walking during load carriage*, *Journal of neuroengineering and rehabilitation* **11** (2014), no. 1 1–11.

[37] A. Tözeren, *Human body dynamics: classical mechanics and human movement*. Springer Science & Business Media, 1999.

[38] D. J. Farris and G. S. Sawicki, *The mechanics and energetics of human walking and running: a joint level perspective*, *Journal of The Royal Society Interface* **9** (2012), no. 66 110–118.

[39] Y. Hurmuzlu and D. B. Marghitu, *Rigid body collisions of planar kinematic chains with multiple contact points*, *The International Journal of Robotics Research (IJRR)* **13** (1994), no. 1 82–92.

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

[41] J. Andersson, *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, KU Leuven, 2013.

[42] A. D. Kuo, *Energetics of actively powered locomotion using the simplest walking model*, *Journal of biomechanical engineering* **124** (2002), no. 1 113–120.

[43] E. R. Westervelt, C. Chevallereau, J. H. Choi, B. Morris, and J. W. Grizzle, *Feedback control of dynamic bipedal robot locomotion*, CRC press, 2007.

[44] C. O. Saglam, *Tractable Quantication of Metastability for Robust Bipedal Locomotion*. PhD thesis, UCSB, 2015.

[45] M. Posa, S. Kuindersma, and R. Tedrake, *Optimization and stabilization of trajectories for constrained dynamical systems*, in *Proc. IEEE Int. Conf. on Robotics and Autom. (ICRA)*, pp. 1366–1373, May, 2016.

[46] S. Sovero, N. Talele, C. Smith, N. Cox, T. Swift, and K. Byl, *Initial data and theory for a high specific-power ankle exoskeleton device*, in *2016 Int. Symp. on Exper. Robotics (ISER)* (D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, eds.), pp. 355–364, Springer, 2017.

[47] G. Bellegarda, N. Talele, and K. Byl, *Nonintuitive optima for dynamic locomotion: The Acrollbot*, pp. 3130–3136, 05, 2018.

[48] E. Todorov, T. Erez, and Y. Tassa, *MuJoCo: A physics engine for model-based control*, in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, Oct, 2012.

[49] N. Talele and K. Byl, *Methods and performance analyses for design and feedback control of efficient and robust planar biped walking*, in *2019 American Control Conference (ACC)*, pp. 4567–4572, July, 2019.

[50] N. Talele and K. Byl, *Mesh-based methods for quantifying and improving robustness of a planar biped model to random push disturbances*, in *2019 American Control Conference (ACC)*, pp. 1860–1866, July, 2019.