AN ABSTRACT OF THE THESIS OF

Mikhail S. Jones for the degree of Master of Science in Mechanical Engineering

presented on June 6, 2014.

Title: Optimal Control of an Underactuated Bipedal Robot

Abstract approved: ______________________________

Jonathan W. Hurst

Bipedal robots represent a unique class of control problems that combine many of the most difficult elements of nonlinear control. These robots are typically designed to be mobile and as such have limited energy and actuator authority making efficiency a prime concern. Unlike wheeled robots, legged robots must transition between different equations of motion as legs make and break contact with the ground, aggravating the complexities of hybrid dynamics. Furthermore, small feet and series elastic actuation of joints leads to an underactuated system, causing traditional nonlinear control methods to struggle.

This thesis describes and validates a general framework for efficiently controlling systems with the properties most problematic in bipedal robots, i.e., nonlinearity, hybrid dynamics, and underactuation. This process is two-step in that, first, an optimal trajectory is generated using direct collocation trajectory optimization, and second, the feasible trajectory is stabilized using a time varying linear quadratic regulator. We demonstrate this process on a number of toy problems including the triple pendulum on a cart and a reduced order template for a running robot. This approach is then applied to the bipedal robot ATRIAS in simulation in order to achieve efficient, dynamic, and robust locomotion behaviors.

Optimal Control of an Underactuated Bipedal Robot

by

Mikhail S. Jones

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 6, 2014
Commencement June 2015

ACKNOWLEDGEMENTS

I would first and foremost like to acknowledge the members of the Dynamic Robotics Laboratory for all of their hard work, friendship and sense of humor over the years. Much of my success is a direct result of their efforts during the many long nights spent debugging the robot. The positive and constructive atmosphere they created was always appreciated.

A special thanks to Christian Hubicki, who first introduced me to the wonderful world of trajectory optimization and who was always there to provide constructive feedback, suggestions and to bounce ideas off of.

Last but not least, a humble thanks to my advisor, Prof. Jonathan Hurst for the continuous support and willingness to let me explore new ideas and hop around between projects as my ever changing interests evolved.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

Chapter 1: Introduction

1.1 Motivation

With countless applications ranging from advancing human prosthetics to aiding in disaster recovery scenarios, legged robots will play a critical part in shaping the future of tomorrow. While the many blockbusters hits would have you believe this day is already upon us, the reality of the matter is we are still a long way from achieving even the most basic of tasks with comparable efficiency, speed, or robustness as animals. This is in large part due to the fully actuated design philosophy adopted by many of todays machines.

Many modern control techniques rely on full actuation to formally stabilize the system dynamics. However, this full actuation approach requires actuators at every joint, which comes with a hidden cost. A revolute joint that was once free spinning, now mounted to motor, suddenly has different dynamics. All of the inertia of the motor and gearbox are now inseparable from the joint's movement. So while controllers can now apply torques, the system is now less dynamic, and moving like a free pivoting joint now costs energy where it previously did not. In legged robots, this means less agility and efficiency; the very quantities we need to maximize. ASIMO, for example, which is likely the most famous humanoid robot, walks in a manner requiring 16 times the energy cost of humans [12].

Underactuated designs on the other-hand show a lot of promise at making robots more efficient and agile, but also throw a lot of traditional control concepts out the window. Typically control theorists use classical methods akin to feedback linearization which override or cancel out dynamics instead of exploiting them. These techniques begin to rip apart at the seams when applied to underactuated systems and only recently have control theorists started more carefully considering the dynamics of these systems and looking for ways to exploit them.

With these ideas guiding my motivation, I seek to use trajectory optimization and stabilization to demonstrate control of underactuated systems in an agile and efficient manner. The ultimate goal being to control ATRIAS, a highly dynamic biped that is also highly underactuated.

1.2 Background

Fully articulated humanoid robots have been the most practical and publicly visible representatives of bipedal locomotion. Notable examples such as Hondas ASIMO [20], AISTs HRP series [23], KAISTs HUBO [27] are electro-mechanically driven, fully actuated machines capable of versatile, autonomous motion carrying their energy source. These high degree of freedom robots address the challenge of bipedal balance by careful regulation of their zero-moment point (ZMP) [34]. However, ensuring controllability of the ZMP calls for actuators and stiff mechanical connections at every joint. This systematic rigidity prevents these humanoids from exhibiting the bouncy, highly dynamic locomotion mastered by animals. This full-actuation approach also tends to consume a lot of power, exhausting on-board batteries in impractically short time spans (estimated under 30 minutes in the case of ASIMO) [12].

Another class of bipedal robots locomote with only little or no actuation utilizing the passive dynamics of the mechanical system. While exhibiting very efficient locomotion, their action is limited to a few gaits and very specific environmental conditions. This class comprises the so called passive dynamic walkers [12] and their motorized offspring, the design of which was driven by the inverted pendulum model for walking [25]. Delfts robots Flame and TUlip are the largest scale implementations of this approach, standing 1.2m tall and weighing 15kg, both were able to walk at 0.45m/s [21]. The most energy economical example is the Cornell Ranger, which bears the pseudo-biped inverted-pendulum configuration common among passive dynamic walkers, but is sufficiently actuated and economical that it has walked over 64km (40 miles) on a single battery charge [6], an ultra-marathon by many definitions.

1.3 Contributions

In this thesis, we seek to develop a clear methodical approach to exploiting the dynamics of underactuated systems and then use this approach to control the ATRIAS bipedal robot. In order to accomplish this goal, we construct a fast and easy-to-use software framework that packages cutting edge techniques for trajectory optimization and stabilization. We demonstrate the use of this tool set on underactuated toy models of increasing complexity. Finally, we generate and stabilize walking gaits on the ATRIAS robot in simulation. The resulting work-optimal gaits demonstrate the fullest embrace of the passive dynamics of this machine.

Chapter 2: Background

2.1 The Optimal Control Problem

Many engineering problems today are driven by the need to find optimal trajectories. These problems are not limited to a single sector but span across a wide range of disciplines including a aeronautics, robotics, economics, and chemistry. Examples include finding optimal orbit transfer trajectories for spacecraft, energy efficient gaits for robots and optimization of chemical reactions to name a few. As different as these problems seem from the surface, they can all be classified as optimal control problems and solved using similar techniques.

Finding an optimal trajectory is only half the battle though, once identified, the candidate trajectory must be stabilized in order to be physically realizable. This two-step process of generating and stabilizing trajectories enables some highly dynamic and unique control measures. In this section we provide a general review of different methods for solving the optimal control problem and stabilizing trajectories. For further details, the reader is referred to [9, 4].

2.1.1 General Formulation

Before proceeding we need to define what an optimal control problem is and how they are formulated. Generally, the goal of an optimal control problem is to determine the state (trajectory) $\boldsymbol{x}(t) = [x_1(t), ..., x_i(t)]^\mathsf{T}$, control input $\boldsymbol{u}(t) = [u_1(t), ..., u_j(t)]^\mathsf{T}$, parameters $\boldsymbol{p} = [p_1(t), ..., p_k(t)]^\mathsf{T}$, initial time t_0, and terminal time t_f that minimizes the scalar performance index

$$J = \Phi(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{p}) + \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}) dt, \tag{2.1}$$

subject to the first order differential equation constraints

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}\right), \tag{2.2}$$

the path constraints

$$\boldsymbol{P}_{min} \leq \boldsymbol{P}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{p}\right) \leq \boldsymbol{P}_{max}, \tag{2.3}$$

and the boundary conditions

$$\phi_{min} \leq \phi\left(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, \boldsymbol{p}\right) \leq \phi_{max}. \tag{2.4}$$

Many real-world systems do not have a single set of continuous dynamics. For example, a running robot transitions from single support dynamics to ballistic dynamics with a discrete jump in between during touchdown impact. When walking, the same robot would transition from single support to double support dynamics. It is therefore necessary to extend the same general formulation used for a single phase system to hybrid systems. Similar to a problem with a single set of dynamics, a hybrid problem consisting of P phases aims to minimize the scalar performance index

$$J = \sum_{i=1}^{P} J^{(i)} \tag{2.5}$$

subject to the first order differential equation constraints

$$\dot{\boldsymbol{x}}^{(i)}(t) = \boldsymbol{f}^{(i)}\left(\boldsymbol{x}^{(i)}(t), \boldsymbol{u}^{(i)}(t), t, \boldsymbol{p}^{(i)}\right), \tag{2.6}$$

the path constraints

$$\boldsymbol{P}^{(i)}_{min} \leq \boldsymbol{P}^{(i)}\left(\boldsymbol{x}^{(i)}(t), \boldsymbol{u}^{(i)}(t), t, \boldsymbol{p}^{(i)}\right) \leq \boldsymbol{P}^{(i)}_{max}, \tag{2.7}$$

the boundary conditions

$$\phi^{(i)}_{min} \leq \phi^{(i)}\left(\boldsymbol{x}^{(i)}(t^{(i)}_0), t^{(i)}_0, \boldsymbol{u}^{(i)}(t^{(i)}_f), t^{(i)}_f, \boldsymbol{p}^{(i)}\right) \leq \phi^{(i)}_{max}, \tag{2.8}$$

and the additional phase linkage constraints

$$L^{(s)}_{min} \leq L^{(s)}\left(\boldsymbol{x}^{(l_s)}(t^{(l_s)}_f), \boldsymbol{u}^{(l_s)}(t^{(l_s)}_f), \boldsymbol{p}^{(l_s)}, t^{(l_s)}_f, \boldsymbol{x}^{(r_s)}(t^{(r_s)}_0), \boldsymbol{u}^{(r_s)}(t^{(r_s)}_0), \boldsymbol{p}^{(r_s)}, t^{(r_s)}_0\right) \leq L^{(s)}_{max}. \tag{2.9}$$

In Equation 2.9, $s = [1, ..., S]$ where S is the total number of linked phase pairs and the vectors r_s and l_s represent lists of linked phase pairs. Typically phases are linked sequentially representing a single continuous motion however it is not uncommon for a trajectory to split (a space craft leaving orbit breaking away from its fuel tanks) or for a trajectory to loop forming periodic orbits (legged robot running gait).

2.2 Solving Optimal Control Problems

Familiar with the general form of optimal control problems, we can now discuss methods to solve them. Most practical or interesting optimal control problems cannot be solved analytically and we must resort to numerical methods to find a solution. These numerical methods can generally be classified as either a direct method or an indirect method. Indirect methods are based on the calculus of variations or the maximum principle and aim to find the root of the necessary condition, $F'(\boldsymbol{x}) = 0$. For an optimal control problem, this requires explicit derivation of the adjoint equations, control equations, and all transversality conditions. Direct methods on the other hand attempt to find the minimum of the objective (or Lagrangian) function. Principally this means a direct method only needs to evaluate the objective function and does not need to explicitly derive or define the necessary conditions.

In practice, indirect methods have several well known drawbacks that make them less desirable than direct methods [4]. The methods are less flexible, minor adjustments to the problem formulation require a user knowledgeable in optimal control theory to re-derive the necessary conditions which can be very difficult for some problems. Additionally, indirect methods are highly sensitive to the initial guess of the adjoint variables which can be very difficult because they have no physical or intuitive meaning. Even with reasonable guesses, the adjoint equations can be ill-conditioned leading to numerical issues. For these reasons, we will focus on direct methods for the remainder of this section.

2.2.1 Direct Transcription Methods

The main idea behind the transcription method is to transcribe a continuous dynamic system into a finite set of variables and constraints and then solve the resulting parameter optimization problem. This process of converting an optimal control problem into a nonlinear programming (NLP) problem is applicable to both direct and indirect methods

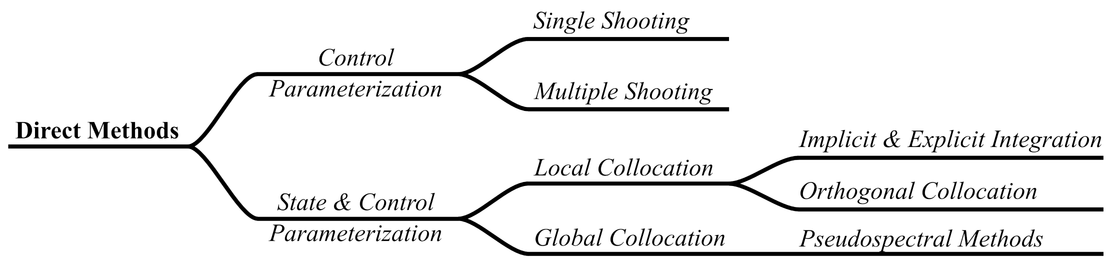

Figure 2.1: Tree diagram showing the hierarchy of various direct transcription methods used for solving optimal control problems. [29]

although we will focus on its applicability to direct methods.

2.2.1.1 Single Shooting Method

The single shooting method is the most intuitive approach to solving a boundary value problem (BVP). The control input is parameterized into some functional form, the dynamics are integrated using a time-marching algorithm, the objective and any constraints are evaluated, and then the control parameters are adjusted (as illustrated in Figure 2.2). This concept is easily understood by considering the simplest example of a cannon being aimed and fired at a target (hence the name single shooting). By adjusting the angle of the cannon, integrating forward through the dynamics, and then evaluating the distance from the target, a solution can readily be found. While simple in concept, the major disadvantage of the single shooting method is that small changes in initial conditions can lead to extreme changes in terminal conditions. This leads to large nonlinearities and poor estimates of the Jacobian matrix increasing the difficulty of finding a reasonable solution.

2.2.1.2 Multiple Shooting Method

An extension of the single shooting method, multiple shooting simply divides the problem into subintervals (shoots) which are then linked together with defect constraints. This increases the size of the resulting nonlinear programming problem as each shoot is now subject to an additional set of initial condition variables and defect constraints. Surpris-

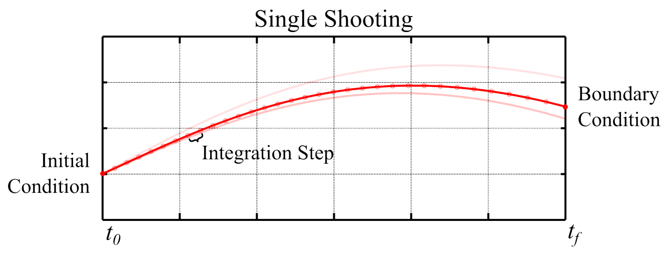

Figure 2.2: Single shooting method for transcribing a optimal controls problem.

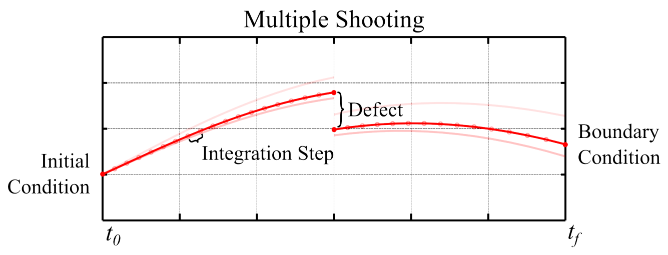

Figure 2.3: Multiple shooting method for transcribing a optimal controls problem.

ingly, this actually reduces the sensitivity and makes the problem substantially easier to solve. This is in part possible because modern sequential quadratic programming (SQP) solvers can take advantage of the inherent sparseness of the Jacobian structure.

2.2.1.3 Direct Collocation Method

Collocation methods take the idea and benefits of multiple shooting a step further, each shoot is reduced down to a single integration step. This enables each defect constraint to be explicitly written out in the form of a single step integration method, popular methods of collocation include Runge-Kutta, Hermite-Simpson, Midpoint, and Euler. Again the resulting Jacobian is highly sparse and large-scale nonlinear programming methods that take advantage of this property excel. By allowing the optimizer direct control over the states and controls while providing guidance via the defect constraints, more information is encoded into the formulation. This can be leveraged by the optimizer, making the problem easier to solve.

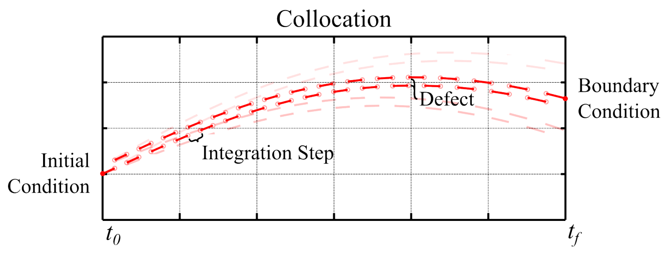

Figure 2.4: Collocation method for transcribing a optimal controls problem.

2.3 Pitfalls of Trajectory Optimization

Trajectory optimization has largely built up a bad reputation and amount of distrust among many researchers. This is largely due to the improper formulation of trajectory optimization problems, overstepping claims, complaints of local minimum, and convergence issues. The real issue is that trajectory optimization is still considered very much an art. While this is somewhat true, many of these issues can be avoided by carefully and thoughtfully formulating problems. Even when being careful the optimizer will always try to cheat the formulation. This could be something subtle that is hardly noticeable such as ringing in the control trajectories or something more physical such as walking upside down underground. The user must be aware of this and be able differentiate the good results from the bad.

That being said, trajectory optimization remains a powerful tool when used properly and the following tips can aid in this process. Defect constraints only hold at the collocation nodes, it is therefore generally a good idea to increase the number of nodes after a solution is found, if the solution does not change then you probably have a good solution. Only locally optimal solutions are found, in many cases this is good enough but starting the problem from different initial conditions can help determine how sensitive the formulation is local minimums. If a dense mesh is required to capture fast dynamics, it is suggested that a less dense mesh is used initially and the solution used as the initial guess to a denser formulation (rinse and repeat as needed).

2.4 Trajectory Stabilization

Executing the optimal control inputs found via trajectory optimization in open-loop does not guarantee stability. In fact, rarely will the resulting trajectories be stable unless stability was explicitly defined as part of the optimization formulation. Small differences in integration step size and accuracy, modeling errors, variations in initial conditions, and disturbances can all cause the system to rapidly diverge in simulation. These effects are only compounded on experimental setups and it is therefore critical that a feedback controller be designed to stabilize the system back to the desired trajectory.

There are many different approaches to trajectory stabilization. Many classical approaches are based on feedback linearization which is fine for fully actuated systems but tends to breakdown when applied to underactuated systems. Because many of the most interesting problems are underactuated systems, we will focus on methods applicable to underactuated systems.

2.4.1 Linear Time Varying (LTV) LQR

A simple yet powerful method for stabilizing trajectories on underactuated systems can be implemented using a slight variation of the Linear Quadratic Regulator (LQR). Most commonly used for stabilizing a system around a fixed point, LQR uses a linear approximation of the dynamics and a quadratic cost function to find the optimal control inputs. In order to apply this to a nonlinear trajectory tracking problem, we consider the following autonomous system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{2.10}$$

Given the feasible trajectories $\boldsymbol{x}^*(t)$ and $\boldsymbol{u}^*(t)$ we perform the coordinate transformation

$$\Delta\boldsymbol{x}(t) = \boldsymbol{x}(t) - \boldsymbol{x}^*(t), \qquad \Delta\boldsymbol{u}(t) = \boldsymbol{u}(t) - \boldsymbol{u}^*(t), \tag{2.11}$$

representing the deviation from the nominal trajectory. Computing the Taylor expansion around this new coordinate system results in the linear time-varying (LTV) system

$$\Delta\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\Delta\boldsymbol{x}(t) + \boldsymbol{b}(t)\Delta\boldsymbol{u}(t) \tag{2.12}$$

where

$$\boldsymbol{A}(t) = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*(t),\boldsymbol{u}^*(t)}, \qquad \boldsymbol{b}(t) = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}^*(t),\boldsymbol{u}^*(t)}. \tag{2.13}$$

The main advantage to doing this is we can now leverage common methods used in linear control, the only difference being that the system is now parameterized as a function of time. We can now formulate a trajectory stabilization problem by penalizing the system for being away from the nominal trajectory resulting in the quadratic cost function

$$J = \Delta\boldsymbol{x}(t_f)^{\mathsf{T}}\boldsymbol{Q}_f\Delta\boldsymbol{x}(t_f) + \int_{t_0}^{t_f}\Big(\Delta\boldsymbol{x}(t)^{\mathsf{T}}\boldsymbol{Q}(t)\Delta\boldsymbol{x}(t) + \Delta\boldsymbol{u}(t)^{\mathsf{T}}\boldsymbol{R}(t)\Delta\boldsymbol{u}(t)\Big)\,dt, \tag{2.14}$$

where

$$\boldsymbol{Q}_f = \boldsymbol{Q}_f^{\mathsf{T}} > \boldsymbol{0}, \qquad \boldsymbol{Q}(t) = \boldsymbol{Q}(t)^{\mathsf{T}} \geq \boldsymbol{0}, \qquad \boldsymbol{R}(t) = \boldsymbol{R}(t)^{\mathsf{T}} > \boldsymbol{0}. \tag{2.15}$$

Notice that the penalty matrices $\boldsymbol{Q}(t)$ and $\boldsymbol{R}(t)$ are time-varying as well although it need not be a requirement. The solution $\boldsymbol{P}(t)$ is computed from reverse-time integration of the continuous time Riccati equation

$$-\dot{\boldsymbol{P}}(t) = \boldsymbol{Q}(t) - \boldsymbol{P}(t)\boldsymbol{b}(t)\boldsymbol{R}^{-1}(t)\boldsymbol{b}^{\mathsf{T}}(t)\boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{A}(t) + \boldsymbol{A}^{\mathsf{T}}(t)\boldsymbol{P}(t), \tag{2.16}$$

with terminal condition

$$\boldsymbol{P}(t_f) = \boldsymbol{Q}_f. \tag{2.17}$$

The choice for the terminal condition $\boldsymbol{Q}_f$ depends on the problem goal. In the case of an infinite-horizon trajectory which moves from state $\boldsymbol{x}(t_0)$ to state $\boldsymbol{x}(t_f)$ and stays there, the terminal condition $\boldsymbol{P}(t_f) = \boldsymbol{P}_\infty$ is preferred where $\boldsymbol{P}_\infty$ is the steady-state solution to the linear time-invariant (LTI) LQR problem at the terminal fixed point. If instead the goal is a periodic orbit, Equation 2.16 can be solved iteratively using $\boldsymbol{P}^{(i)}(t_f) = \boldsymbol{P}^{(i-1)}(t_0)$ until the solution converges.

Once solved, the optimal control policy represented by a time varying gain matrix can be computed as

$$\boldsymbol{k}(t) = \boldsymbol{R}(t)^{-1}\boldsymbol{b}^{\mathsf{T}}(t)\boldsymbol{P}(t), \tag{2.18}$$

and the feedback control law can be formulated as

$$\boldsymbol{u}(t) = \boldsymbol{u}^*(t) + \boldsymbol{k}^{\mathsf{T}}(t)(\boldsymbol{x}^*(t) - \boldsymbol{x}(t)). \tag{2.19}$$

Chapter 3: Methods

In this chapter we describe our methods for generating and stabilizing optimal control trajectories for underactuated systems. This lays the framework which will then be used to control an underactuated bipedal robot. To achieve this, we created an object-oriented software package for the fast, convenient, and reliable generation of optimal trajectories. We benchmarked the tool on classic optimal control problems with varying degrees of linearity and underactuation. Verifying computation speed and reliability is important for the target task, controlling a walking robot, which has many state variables and degrees of underactuation.

3.1 COALESCE

Countless software packages have already been developed to solve optimal control problems using direct transcription methods (SOCS [5], DIRCOL [33], OTIS [19], GPOPS-II [28], and JModelica [1]). These packages all suffer from some common drawbacks that restrict the typical users ability to efficiently or effectively solve problems.

- Implemented in low-level languages with poor APIs resulting in cumbersome, difficult, and error-prone interfacing.
- Require the user to provide the objective and/or gradient functions which require extensive knowledge of optimal control methods.
- Restricted to a specific optimization algorithm which may not work for all cases.

To overcome these drawbacks we designed COALESCE to provide an efficient, versatile, and seamless interface for formulating and solving optimal control problems. In order to achieve this, COALESCE is built in *MATLAB* [24] using an object-oriented interface to abstract the problem formulation, allowing for a simple, intuitive and robust problem generation. Additionally, by automatically generating analytical gradients and exporting the formulated function scripts, greater performance and accuracy can be achieved with no additional user overhead. This greatly increases the non-expert user's ability to formulate and solve problems with minimal headaches or errors.

3.1.1 Features

COALESCE has many built-in features that allow users with only a basic knowledge or background in numerical optimization and direct collocation methods to quickly start formulating and solving problems.

- Computes all constraint and objective gradients and corresponding sparsity patterns analytically.
- Exports optimized functions by simplifying equations, substituting constants, and using sparse matrices. Can additionally export low-level code to compile functions to MEX files.
- Interfaces with the MATLAB Optimization Toolbox as well as other popular solvers including SNOPT [14], IPOPT [35], and KNITRO [10]. No additional user overhead is required to export and solve with another solver.
- Provides additional functionality for visualizing solutions, identifying infeasible constraints, automatically encoding collocation equation constraints and exporting results.
- Supports multi-phase optimal control problems using various implicit and explicit direct collocation methods.
- Programmed in a high-level language with object oriented code allows additional functionality to be added on with relative ease.

3.1.2 Implementation

At the heart of the COALESCE package is the base *Problem* class which solves a standard nonlinear programming problem. Given the decision variables $\boldsymbol{x} = [x_1, ..., x_i]^\mathsf{T}$ and parameters $\boldsymbol{p} = [p_1, ..., p_j]^\mathsf{T}$, minimize the scalar objective function

$$f(\boldsymbol{x}, \boldsymbol{p}) \tag{3.1}$$

subject to

$$\boldsymbol{c}_l \leq \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{p}) \leq \boldsymbol{c}_u \tag{3.2}$$

where $\boldsymbol{c}_l$ and $\boldsymbol{c}_u$ are the lower and upper bounds, respectively, of the nonlinear constraints $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{p})$. After initializing, *Parameter*, *Variable*, *Objective*, and *Constraint* objects are then added to the *Problem* object. The then formulated problem is passed to the abstract *Solver* class which exports and solves the problem using the specified solver. This allows the user to quickly and easily change solvers without having to reformulate the problem to match the solver API. The resulting solution is propagated back through the *Problem* class structure allowing easy access to variable solutions.

Optimal control problems can be formulated by using the *Phase* class which simply builds on top of the *Problem* class. *State* and *Control* objects are added which automatically construct the necessary *Parameter*, *Variable*, and *Constraint* objects to describe the direct collocation formulation. Multiple phase can be linked together using the *PhaseSchedule* class keeping all basic functionality associated with the *Problem* class. Again, once solved the solution is propagated back through the class structure so the results can be analyzed.

3.2 Optimal Control Examples

In order to demonstrate these concepts and benchmark different methods, toy problems of increasing complexity are formulated and solved in this section. All problems were transcribed into a nonlinear programming (NLP) problem using the developed software package COALESCE (Section 3.1). The resulting optimization problem was solved using the sparse nonlinear sequential quadratic programming (SQP) solver SNOPT [14].

3.2.1 Double Integrator

We begin with the simplest possible linear second order system, the double integrator. Intuitively, the double integrator can be thought of as a unit mass sliding on an frictionless surface due to a single horizontal external force. A common optimal control problem used for demonstrative purposes seeks to translate this mass from some initial position to a target position in the least amount of time possible. This is analagous to driving an ideal car from point A to point B as fast as possible subject to velocity and acceleration

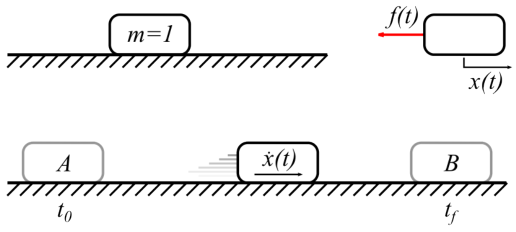

Figure 3.1: The double integrator (or sliding mass) is a classic optimal control problem with linear continuous dynamics. The system model and generalized coordinates are depicted on the top, while the minimum time optimal control problem is illustrated on the bottom.

limits.

3.2.1.1 System Model

The double integrator $\ddot{x} = u$ can be represented by the linear time invariant (LTI) system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{3.3}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 1 \end{bmatrix}. \tag{3.4}$$

Furthermore, the system is subject to the following velocity and force limitations

$$|\dot{x}| \leq 1.0 m/s, \qquad |f| \leq 1.0 N. \tag{3.5}$$

3.2.1.2 Minimum Time Problem

The objective of the minimum time double integrator problem is to move from the stationary point given by initial conditions

$$\mathbf{x}(0) = 0m, \qquad \dot{\mathbf{x}}(0) = 0m/s \tag{3.6}$$

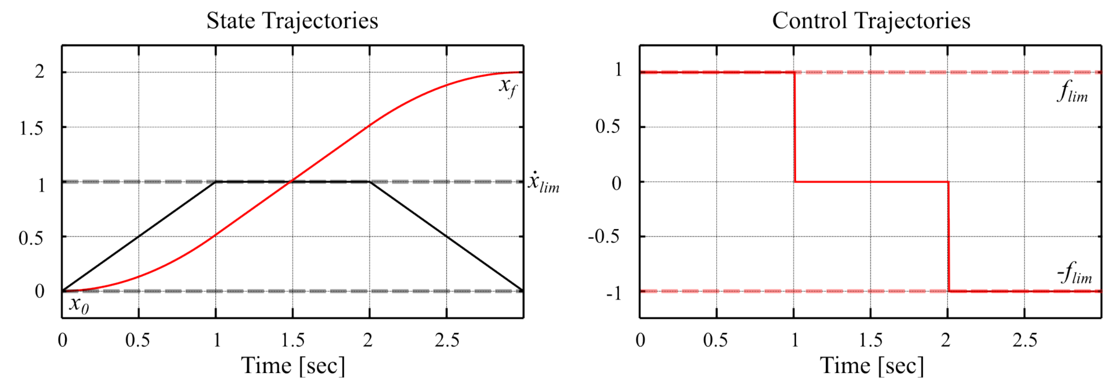

Figure 3.2: Optimal state and control trajectories for the minimum time double integrator optimal control problem. The optimal control trajectory illustrates bang-bang control, the known analytic solution to the minimum time problem.

to another stationary point given by

$$\mathbf{x}(T) = 2m, \qquad \dot{\mathbf{x}}(T) = 0m/s \tag{3.7}$$

while minimizing the total maneuver duration T.

3.2.1.3 Trajectory Optimization

The resulting optimal trajectories (Figure 3.2) illustrate bang-bang control. Intuitively we know this to be correct, the fastest way to get somewhere is to accelerate as fast as you can for as long as you can before rapidly decelerating. Because of simplicity of the model, this can of course be proven analytically and it is well know that the solution to the minimum time double integrator probelm is bang-bang control.

3.2.2 Double Pendulum on a Cart

The single pendulum on a cart (sometimes referred to as the cart-pole problem) is a classic example of an underactuated nonlinear controls problem. The system has been widely used to demonstrate and benchmark nonlinear control techniques by stabilizing the pendulum around an unstable fixed point (inverted pendulum) [26, 2]. Further-

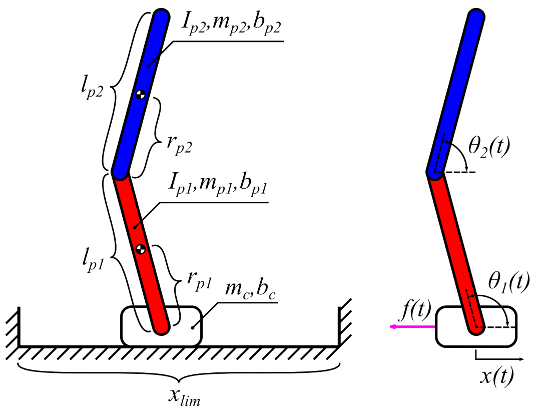

Figure 3.3: An extension of the classic cart-pole control problem, the double pendulum on a cart swing-up and side-step problems are significantly more complex due to the extra degree of underactuation. The system model and generalized coordinates for the double pendulum on a cart are illustrated above.

more, the system has been used to investigate the swing-up and side-stepping optimal control problems [3, 15, 17]. Recently, these problems have gained increasing popularity and variations have been addressed in research literature for the acrobot [31, 36], pendubot [32, 13], and double pendulum on a cart [18, 30].

3.2.2.1 System Model

The double pendulum on a cart (Figure 3.3) consists of three rigid bodies; an actuated cart constrained to move along a horizontal track, and two pendula connected to the cart in series through pin joints. The system is controlled by a single force acting on the cart while all other degrees of freedom remain unactuated. The system parameters and corresponding values are summarized in Table 3.1. Furthermore, the system is subject to the following physical limitations

$$|x| \leq 1.0m, \qquad |f| \leq 10.0N. \tag{3.8}$$

The differential equations of motion for the double pendulum on a cart can be derived using the Lagrange method. The forward kinematic relations describing the center of

Parameter	Description	Value
m_c	Cart Mass	$1.0\,[kg]$
b_c	Cart Friction	$0.5\,[N\,s/m]$
I_{p1}, I_{p2}	Pendula Inertia	$0.01\,[kg\,m^2]$
m_{p1}, m_{p2}	Pendula Mass	$0.01\,[kg]$
b_{p1}, b_{p2}	Pendula Friction	$0.005\,[N\,m\,s/rad]$
r_{p1}, r_{p2}	Pendula Distance to COM	$0.25\,[m]$
l_{p1}, l_{p2}	Pendula Length	$0.5\,[m]$

Table 3.1: List of double pendulum on a cart model parameters.

mass locations in terms of our generalized coordinates $\boldsymbol{q} = [x, \theta_1, \theta_2]^\mathsf{T}$ are given by

$$\boldsymbol{p}_c = \begin{bmatrix} x \\ 0 \end{bmatrix}, \quad \boldsymbol{p}_{p1} = \begin{bmatrix} x + r_{p1}\cos(\theta_1) \\ r_{p1}\sin(\theta_1) \end{bmatrix}, \quad \boldsymbol{p}_{p2} = \begin{bmatrix} x + l_{p1}\cos(\theta_1) + r_{p2}\cos(\theta_2) \\ l_{p1}\sin(\theta_1) + r_{p2}\sin(\theta_2) \end{bmatrix}. \tag{3.9}$$

Thus, the kinetic energy in the system is

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2} m_c \dot{\boldsymbol{p}}_c^\mathsf{T} \dot{\boldsymbol{p}}_c + \frac{1}{2} \sum_{i=1}^{2} \left[m_{pi} \dot{\boldsymbol{p}}_{pi}^\mathsf{T} \dot{\boldsymbol{p}}_{pi} + I_{pi} \dot{\theta}_i^2 \right] \tag{3.10}$$

and the potential energy due to the gravitational field g is given by

$$V(\boldsymbol{q}) = g \sum_{i=1}^{2} [m_{pi} \boldsymbol{p}_{pi,2}] \tag{3.11}$$

From Hamilton's principle, the equations of motion can be computed from the Lagrangian $\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = T(\boldsymbol{q}, \dot{\boldsymbol{q}}) - V(\boldsymbol{q})$ as

$$\frac{d}{dt} \frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}_i} - \frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \boldsymbol{q}_i} + \frac{\partial \mathcal{R}(\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}_i} = \boldsymbol{\tau}_i, \qquad i = 1, ..., 3 \tag{3.12}$$

where $\boldsymbol{\tau}_i = [f \;\; \boldsymbol{0}_{1\times 2}]^\mathsf{T}$ are the external forces acting on each i^{th} generalized coordinate and $\mathcal{R}(\dot{\boldsymbol{q}})$ is the Rayleigh dissipation function defined as

$$\mathcal{R}(\dot{\boldsymbol{q}}) = \frac{1}{2} b_c \dot{x}^2 + \frac{1}{2} b_{p1} \dot{\theta}_1^2 + \frac{1}{2} b_{p2} (\dot{\theta}_2 - \dot{\theta}_1)^2 \tag{3.13}$$

$$\boldsymbol{H} = \begin{bmatrix} m_c + m_{p1} + m_{p2} & -\sin(\theta_1)(l_{p1}m_{p2} + m_{p1}r_{p1}) & -m_{p2}r_{p2}\sin(\theta_1) \\ -\sin(\theta_1)(l_{p1}m_{p2} + m_{p1}r_{p1}) & m_{p2}l_{p1}^2 + m_{p1}r_{p1}^2 + I_{p1} & l_{p1}m_{p2}r_{p2}cos(\theta_1 - \theta_2) \\ -m_{p2}r_{p2}\sin(\theta_2) & l_{p1}m_{p2}r_{p2}\cos(\theta_1 - \theta_2) & m_{p2}r_{p2}^2 + I_{p2} \end{bmatrix}$$

$$\begin{aligned} \boldsymbol{\eta}_1 &= f - \dot{x}b_c + m_{p2}l_{p1}\cos(\theta_1)\dot{\theta}_1^2 + m_{p2}r_{p2}\cos(\theta_2)\dot{\theta}_2^2 + \dot{\theta}_1^2 m_{p1}r_{p1}\cos(\theta_1) \\ \boldsymbol{\eta}_2 &= -l_{p1}m_{p2}r_{p2}\sin(\theta_1 - \theta_2)\dot{\theta}_2^2 + b_{p2}\dot{\theta}_2 - \dot{\theta}_1 b_{p1} - \dot{\theta}_1 b_{p2} - g l_{p1}m_{p2}\cos(\theta_1) - g m_{p1}r_{p1}\cos(\theta_1) \\ \boldsymbol{\eta}_3 &= l_{p1}m_{p2}r_{p2}\sin(\theta_1 - \theta_2)\dot{\theta}_1^2 + b_{p2}\dot{\theta}_1 - \dot{\theta}_2 b_{p2} - g m_{p2}r_{p2}\cos(\theta_2) \end{aligned}$$

Table 3.2: Inertia matrix $\boldsymbol{H}(\boldsymbol{q})$ and nonconservative force vector $\boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$ for the double pendulum on a cart equations of motion.

The equations of motion can then be written in general matrix form

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} = \boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}) \tag{3.14}$$

where matrix $\boldsymbol{H}(\boldsymbol{q})$ and vector $\boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$ are defined in Table 3.2. Because the inertia matrix $\boldsymbol{H}(\boldsymbol{q})$ is positive definite and thus invertible, we can rewrite Equation (3.14) in terms of acceleration and by then introducing the new state vector $\boldsymbol{x} = [\boldsymbol{q}\ \dot{\boldsymbol{q}}]^{\mathsf{T}}$, reduce to the first order system of equations defined as

$$\boldsymbol{f}(\boldsymbol{x}, u) = \dot{\boldsymbol{x}} = \frac{d}{dt}\begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H}^{-1}(\boldsymbol{q})\boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}) \end{bmatrix}. \tag{3.15}$$

3.2.2.2 Minimum Time Swing-Up Problem

The objective of the minimum time swing-up problem is to swing the pendula from the downward hanging stable fixed point state given by initial conditions

$$\boldsymbol{q}(0) = [0, -\pi/2, -\pi/2]^{\mathsf{T}}, \qquad \dot{\boldsymbol{q}}(0) = \boldsymbol{0} \tag{3.16}$$

to the inverted unstable fixed point state given by

$$\boldsymbol{q}(T) = [0, \pi/2, \pi/2]^{\mathsf{T}}, \qquad \dot{\boldsymbol{q}}(T) = \boldsymbol{0} \tag{3.17}$$

while minimizing the total maneuver duration T.

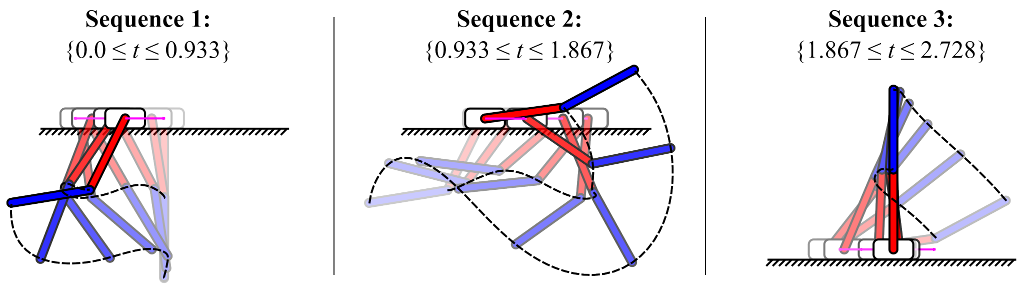
Sequence 1:
$\{0.0 \le t \le 0.933\}$
Sequence 2:
$\{0.933 \le t \le 1.867\}$
Sequence 3:
$\{1.867 \le t \le 2.728\}$

Figure 3.4: Stroboscopic illustration of the time optimal double pendulum on a cart swing-up maneuver. For each sequence the darkness increases with time.

3.2.2.3 Minimum Time Side-Stepping Problem

The objective of the minimum time side-stepping problem is to balance the pendula at the inverted unstable fixed point while translating the cart from one state given by initial conditions

$$\boldsymbol{q}(0) = [-3/2, \pi/2, \pi/2]^{\mathsf{T}}, \qquad \dot{\boldsymbol{q}}(0) = \mathbf{0} \tag{3.18}$$

to another state given by

$$\boldsymbol{q}(T) = [3/2, \pi/2, \pi/2]^{\mathsf{T}}, \qquad \dot{\boldsymbol{q}}(T) = \mathbf{0} \tag{3.19}$$

while minimizing the total maneuver duration T.

3.2.2.4 Trajectory Optimization

The resulting time optimal maneuvers are illustrated in Figures 3.4 and 3.5. Using only the feed forward control input, the system was re-simulated using an adaptive time step integrator and the corresponding open-loop state time histories are compared to the nominal trajectories in Figures 3.6 and 3.8.

3.2.2.5 Feedback Control Design

The importance of feedback control becomes readily apparent when observing the open-loop time histories (Figures 3.6 and 3.8). While the swing-up maneuver open-loop re-

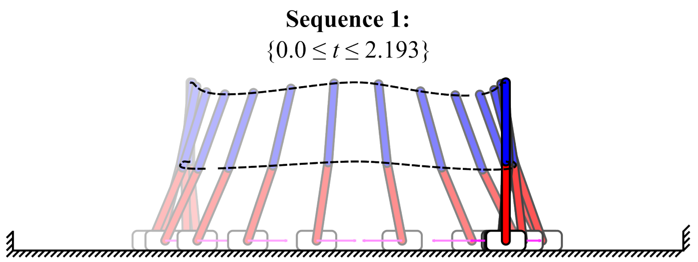

Figure 3.5: Stroboscopic illustration of the time optimal double pendulum on a cart side-step maneuver. For each sequence the darkness increases with time.

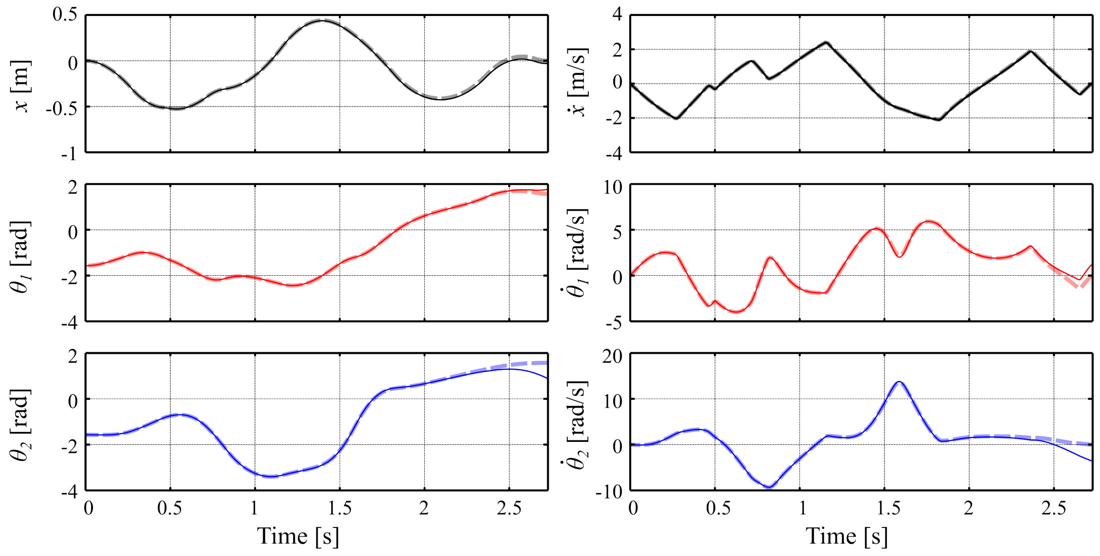

Figure 3.6: The computed optimal trajectories (lighter dashed lines) are compared to the simulated open-loop trajectories (thin solid lines) for the double pendulum on a cart time optimal swing-up maneuver. The open-loop response performs very well and approaches the target terminal condition.

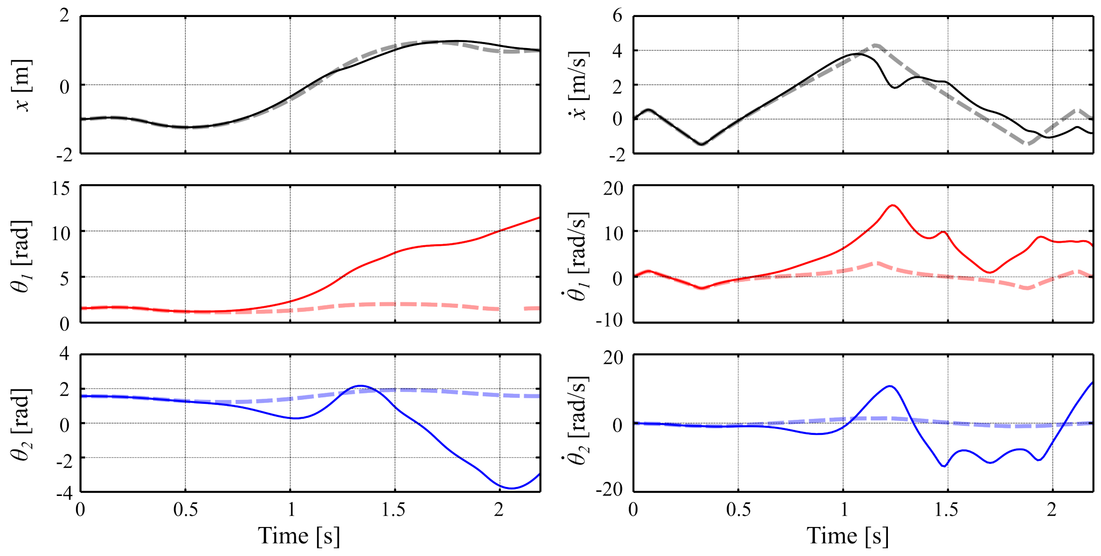

Figure 3.7: The computed optimal trajectories (lighter dashed lines) are compared to the simulated open-loop trajectories (thin solid lines) for the double pendulum on a cart time optimal side-step maneuver. The open-loop response quickly diverges from the target trajectory highlighting the need for a feedback controller to regulate the optimized trajectory.

sponse approaches the target terminal condition, the side-stepping maneuver quickly diverges from the desired trajectory. This may be surprising initially but intuitively it makes sense. The swing up maneuver starts from the stable fixed point with the minimum potential energy and so stabilization is not an issue until later in the maneuver. The side-stepping maneuver on the other-hand starts from an unstable fixed point with the maximum amount of potential energy and thus requires feedback stabilization from the start. It is interesting to note that while stabilization of an inverted double pendulum is a relatively complex task, the optimizer doesn't concern itself with stabilization and actually finds a solution much quicker than for the swing-up maneuver. This just emphasizes the importance of using trajectory stabilization along side trajectory optimization.

In order to stabilize either maneuver back to the nominal trajectory in presence of numerical error, disturbances, and even initial condition offsets, a linear time-varying (LTV) linear quadratic regulator (LQR) was derived and implemented. For clarity and to keep this chapter self contained, some equations from Section 2.4.1 are repeated. Linearizing the equations of motion (Equation 3.15) around a coordinate system that moves along the feasible trajectories $\boldsymbol{x}^*(t)$ and $u^*(t)$ results in the time-varying linear system

$$\Delta\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\Delta\boldsymbol{x}(t) + \boldsymbol{b}(t)\Delta u(t) \tag{3.20}$$

where

$$\boldsymbol{A}(t) = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, u)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^*(t), u^*(t)}, \qquad \boldsymbol{b}(t) = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, u)}{\partial u}\right|_{\boldsymbol{x}^*(t), u^*(t)}, \tag{3.21}$$

and

$$\Delta\boldsymbol{x}(t) = \boldsymbol{x}(t) - \boldsymbol{x}^*(t), \qquad \Delta u(t) = u(t) - u^*(t). \tag{3.22}$$

Given this time-varying linear system, a linear quadratic regulator can be derived that minimizes the cost function given by

$$J = \Delta\boldsymbol{x}(T)^{\mathsf{T}}\boldsymbol{Q}_f\Delta\boldsymbol{x}(T) + \int_0^T \Big(\Delta\boldsymbol{x}(t)^{\mathsf{T}}\boldsymbol{Q}\Delta\boldsymbol{x}(t) + \Delta u(t)^{\mathsf{T}}\boldsymbol{R}\Delta u(t)\Big)\,dt. \tag{3.23}$$

The time-varying gain matrix $\boldsymbol{k}(t)$ is computed from reverse-time integration of the

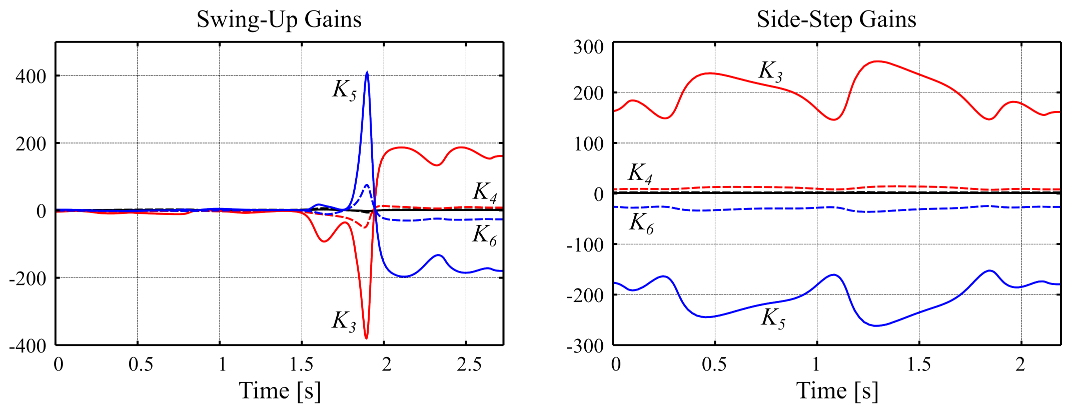

Figure 3.8: Time-varying linear quadratic regulator gains $\boldsymbol{K}_i(t), i = 1, ..., 6$ for the double pendulum on a cart swing-up (left side) and side-step (right side) maneuvers. For both maneuvers, the gains associated with cart position and velocity error ($\boldsymbol{K}_1(t)$ and $\boldsymbol{K}_2(t)$) are not labeled because they are so small relative to the other gains.

Riccati equation

$$-\dot{\boldsymbol{P}}(t) = \boldsymbol{Q} - \boldsymbol{P}(t)\boldsymbol{b}(t)\boldsymbol{R}^{-1}\boldsymbol{b}^{\mathsf{T}}(t)\boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{A}(t) + \boldsymbol{A}^{\mathsf{T}}(t)\boldsymbol{P}(t), \qquad \boldsymbol{P}(T) = \boldsymbol{Q}_f \quad (3.24)$$

where

$$\boldsymbol{k}(t) = \boldsymbol{R}^{-1}\boldsymbol{b}^{\mathsf{T}}(t)\boldsymbol{P}(t). \quad (3.25)$$

Since we want to stabilize an infinite-horizon trajectory, we select the boundary condition $\boldsymbol{P}(T) = \boldsymbol{P}_\infty$ where $\boldsymbol{P}_\infty$ is the steady-state solution to the linear time-invariant (LTI) LQR problem at the terminal fixed point. The resulting time-varying gain matrices depicted in Figure 3.8 were computed using the following weighting matrices

$$\boldsymbol{Q} = diag(1, 1, 1, 1, 1, 1), \qquad \boldsymbol{R} = 1. \quad (3.26)$$

Using the feedback controller given by

$$u(t) = u^*(t) + \boldsymbol{k}^{\mathsf{T}}(t)(\boldsymbol{x}^*(t) - \boldsymbol{x}(t)) \quad (3.27)$$

both maneuvers were then simulated from perturbed initial conditions. The resulting stabilized trajectories are shown in Figures 3.9 and 3.10. Even with the simple identity

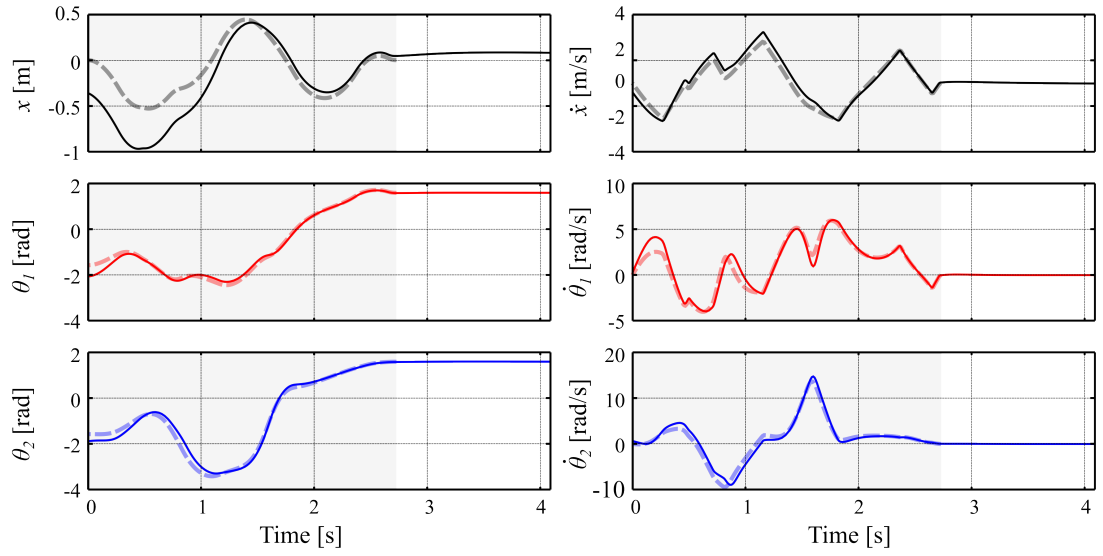

Figure 3.9: The computed optimal trajectories (lighter dashed lines) are compared to the simulated closed-loop trajectories (thin solid lines) for the double pendulum on a cart time optimal swing-up maneuver. Even with significantly perturbed initial conditions, the controller is able to "catch-up" with the target trajectory and stabilize the system.

weighting matrices the system is robust enough to recover from significant disturbances in initial conditions while respecting actuator limits. However, because the nominal control input is bang-bang, there is an inherent lack of control authority to stabilize the system in certain directions. When disturbed in these directions the system has no chance to recover. This could be dealt with by finding an time optimal trajectory that uses an actuator limits less than the actual limit. The lesser the limit, the more overhead available for making corrections.

3.2.3 Triple Pendulum on a Cart

Continuing with the pendulum on a cart theme, we now extend the problem and examine the lesser investigated problem of the triple pendulum on a cart. While successful swing-up and side-stepping of a triple pendulum on cart have been achieved experimentally [15, 17], the maneuvers were discovered by solving a boundary value problem (BVP) with a fixed duration.

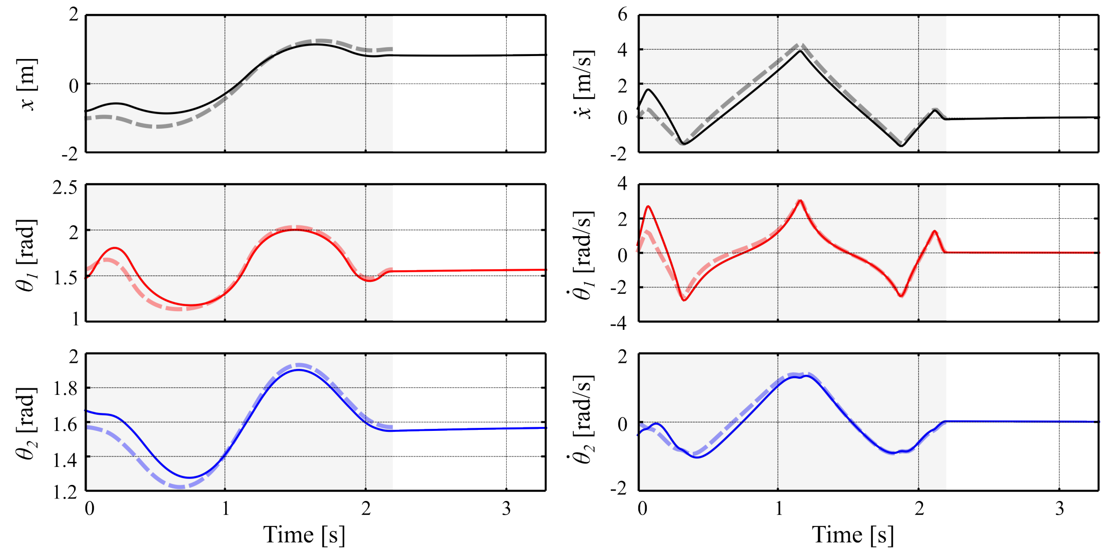

Figure 3.10: The computed optimal trajectories (lighter dashed lines) are compared to the simulated closed-loop trajectories (thin solid lines) for the double pendulum on a cart time optimal side-step maneuver. Because the side-step maneuver starts from an unstable fixed point it is more susceptible to initial condition perturbations than the swing up maneuver. Despite this, the controller is able to regulate the system back to the nominal trajectory for moderate offsets.

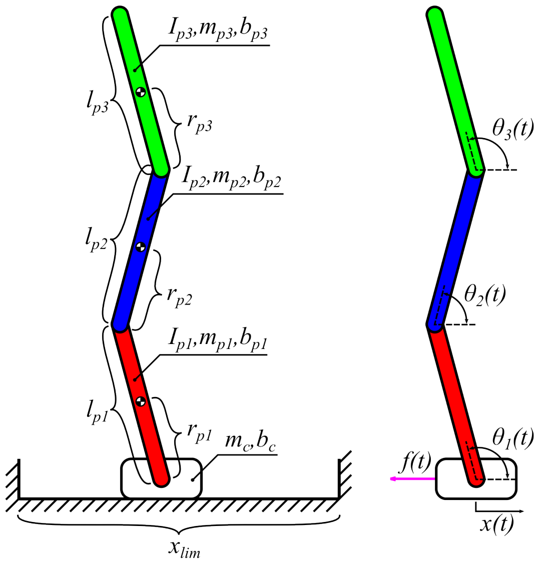

Figure 3.11: The lesser studied triple pendulum on a cart swing-up and side-step maneuvers are significantly more difficult due to the extra degree of underactuation. The system model and generalized coordinates are illustrated above.

3.2.3.1 System Model

The triple pendulum on a cart (Figure 3.11) consists of three rigid bodies; an actuated cart constrained to move along a horizontal track, and three pendula connected to the cart in series through pin joints. The system is controlled by a single force acting on the cart while all other degrees of freedom remain unactuated. The system parameters are summarized in Table 3.3. Furthermore, the system is subject to the following physical limitations

$$|x| \leq 1.0m, \qquad |f| \leq 10.0N. \tag{3.28}$$

Using the same process as the double pendulum on a cart, but with an additional link, the differential equations of motion for the triple pendulum on a cart are derived using the Lagrange method. The resulting first order system of equations is written as

$$\boldsymbol{f}(\boldsymbol{x}, u) = \dot{\boldsymbol{x}} = \frac{d}{dt}\begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H}^{-1}(\boldsymbol{q})\boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}) \end{bmatrix}. \tag{3.29}$$

Parameter	Description	Value
m_c	Cart Mass	$1.0\,[kg]$
b_c	Cart Friction	$0.5\,[N\,s/m]$
I_{p1}, I_{p2}, I_{p3}	Pendula Inertia	$0.01\,[kg\,m^2]$
m_{p1}, m_{p2}, m_{p3}	Pendula Mass	$0.01\,[kg]$
b_{p1}, b_{p2}, b_{p3}	Pendula Friction	$0.005\,[N\,m\,s/rad]$
r_{p1}, r_{p2}, r_{p3}	Pendula Distance to COM	$0.25\,[m]$
l_{p1}, l_{p2}, l_{p3}	Pendula Length	$0.5\,[m]$

Table 3.3: List of triple pendulum on a cart model parameters.

where $\boldsymbol{H}(\boldsymbol{q})$ is the inertia matrix, vector $\boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$ is the left over Coriolis and gravity terms, and $\boldsymbol{q} = [x, \theta_1, \theta_2, \theta_3]^\mathsf{T}$ is the generalized coordinates.

3.2.3.2 Minimum Time Swing-Up Problem

The objective of the minimum time swing-up problem is to swing the pendula from the downward hanging stable fixed point state given by initial conditions

$$\boldsymbol{q}(0) = [0, -\pi/2, -\pi/2, -\pi/2]^\mathsf{T}, \qquad \dot{\boldsymbol{q}}(0) = \boldsymbol{0} \tag{3.30}$$

to the inverted unstable fixed point state given by

$$\boldsymbol{q}(T) = [0, \pi/2, \pi/2, \pi/2]^\mathsf{T}, \qquad \dot{\boldsymbol{q}}(T) = \boldsymbol{0} \tag{3.31}$$

while minimizing the total maneuver duration T.

3.2.3.3 Minimum Time Side-Stepping Problem

The objective of the minimum time side-stepping problem is to balance the pendula at the inverted unstable fixed point while translating the cart from one state given by initial conditions

$$\boldsymbol{q}(0) = [-3/2, \pi/2, \pi/2, \pi/2]^\mathsf{T}, \qquad \dot{\boldsymbol{q}}(0) = \boldsymbol{0} \tag{3.32}$$

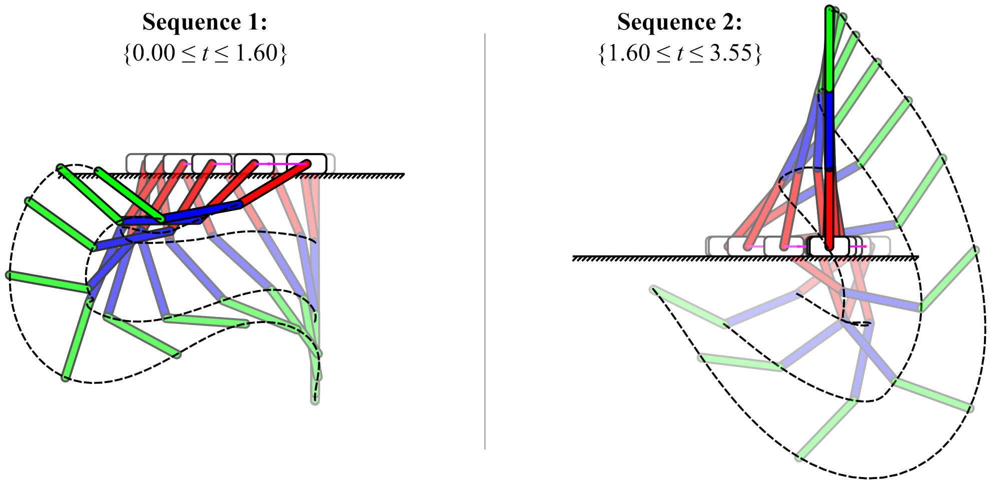

Figure 3.12: Stroboscopic illustration of the time optimal triple pendulum on a cart swing-up maneuver. For each sequence the darkness increases with time.

to another state given by

$$\boldsymbol{q}(T) = [3/2, \pi/2, \pi/2, \pi/2]^\mathsf{T}, \qquad \dot{\boldsymbol{q}}(T) = \mathbf{0} \tag{3.33}$$

while minimizing the total maneuver duration T.

3.2.3.4 Trajectory Optimization

The discovered time optimal swing-up and side-step maneuvers are illustrated in Figures 3.12 and 3.13.

3.2.3.5 Feedback Control Design

Using the same methodology as the double pendulum on a cart, a linear time varying linear quadratic regulator is designed to stabilize around the nominal trajectory. Despite the extra degree of underactuation the regulator performs surprisingly well and is able to stabilize the maneuvers from substantial offsets in initial conditions. However, because the controller is time dependent, the system will occasionally fail from perturbations which cause the system to get ahead of its self. In other words, if the perturbation

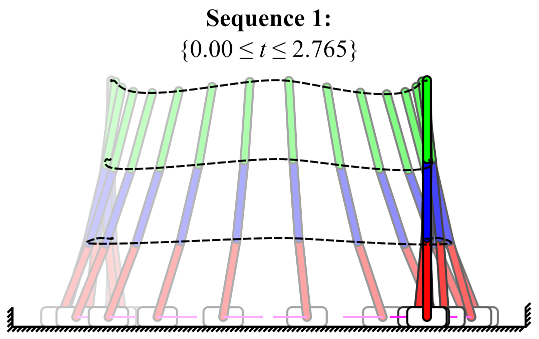

Figure 3.13: Stroboscopic illustration of the time optimal triple pendulum on a cart side-step maneuver. For each sequence the darkness increases with time.

pushes in the direction of the desired state progression, the controller will try to resist that movement which can have catastrophic results especially if it's during a dynamic or passive region of the trajectory.

3.2.4 Reduced Order Gait Generation

Reduced order dynamic models are commonly used to encapsulate the most important properties and behaviors of complex systems while retaining physical relevance. This enables more general conclusions to be formed from the resulting insights. Common reduced order locomotion models include the compass gait walker [16], spring loaded inverted pendulum (SLIP) model [7], and the rimless wheel [11]. While these models are low-dimensional, they are inherently still complex as locomotion dynamics are nonlinear, have discrete and continuous regions resulting from impacts, and are heavily underactuated.

3.2.4.1 System Model

The popular and highly relevant model used in this section is commonly referred to as the spring-mass model. The SLIP model has been shown to approximate animal walking and running behaviors in everything from cockroaches, to quail, to kangaroos [8], to humans [7]. This makes it an ideal candidate model to use for the investigation of efficient locomotion behaviors. As an added benefit, the robot used in Chapter 4 is

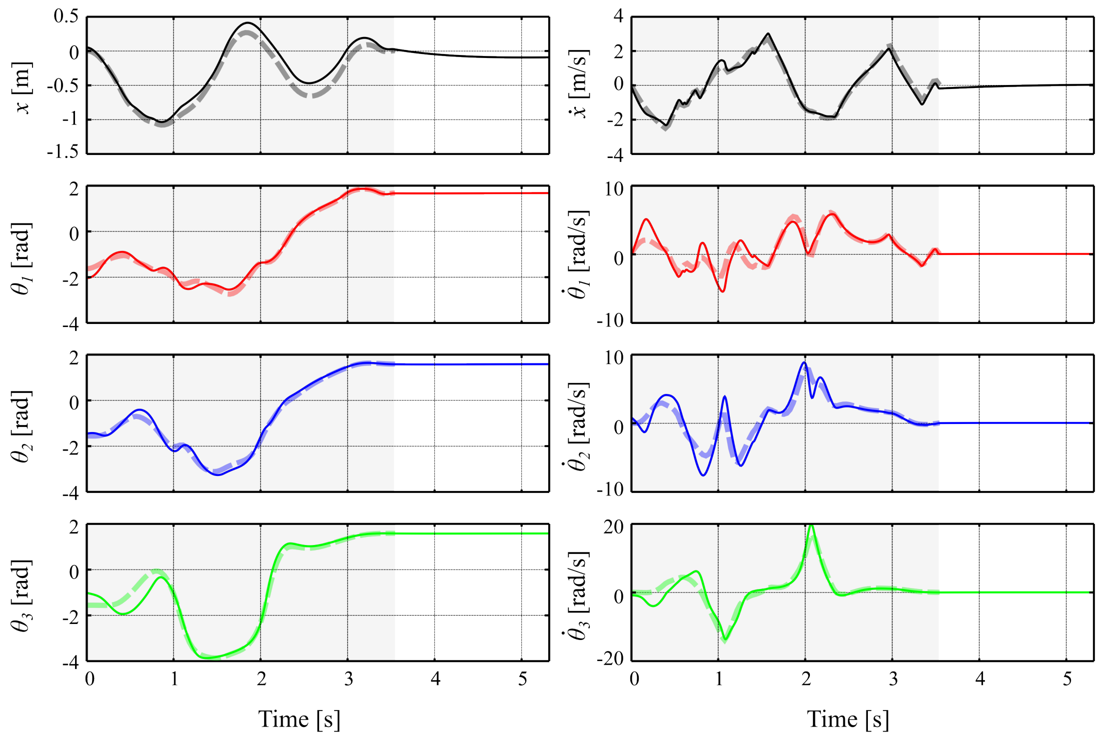

Figure 3.14: The computed optimal trajectories (lighter dashed lines) are compared to the simulated closed-loop trajectories (thin solid lines) for the triple pendulum on a cart time optimal swing-up maneuver. Disturbances on the pendula states can only be corrected with large cart movements thus the cart state tends to diverge slightly returning to the target after the pendulum is inverted.

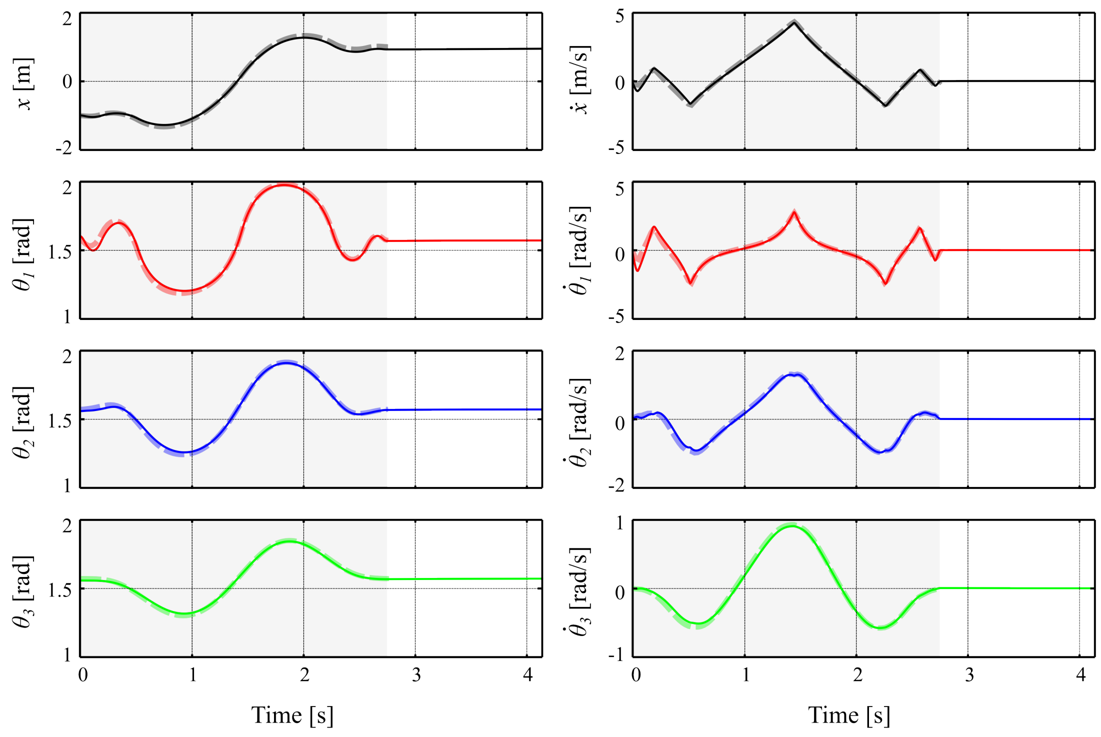

Figure 3.15: The computed optimal trajectories (lighter dashed lines) are compared to the simulated closed-loop trajectories (thin solid lines) for the triple pendulum on a cart time optimal side-step maneuver. The stabilization of a triple pendulum is much more sensitive to disturbances than an double pendulum making a time optimal side step very challenging.

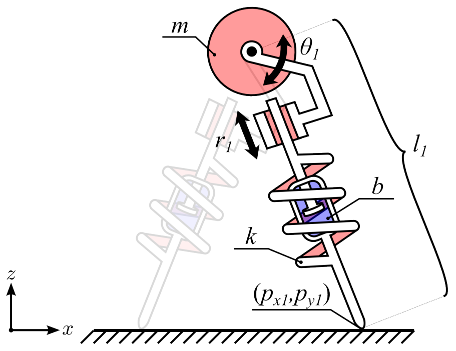

Figure 3.16: The spring loaded inverted pendulum (SLIP) model is a popular reduced order template used to express animal walking and running behaviors. The system model illustrated above is variation of the SLIP model that includes actuators with series springs and dampers introducing inherent losses.

based on this exact model.

The spring-mass model consists of a point mass with two massless springy legs. We use a variation of this model (Figure 3.16) that includes actuators and damping in the leg length direction, thus introducing inherent system losses. The equations of motion for the system during double support can readily be produced using the Newtonian method and summing the forces resulting in

$$\ddot{x} = \frac{f_1 \cos(\theta_1) + f_2 \cos(\theta_2)}{m} \tag{3.34}$$

$$\ddot{z} = \frac{f_1 \sin(\theta_1) + f_2 \sin(\theta_2)}{m} - g \tag{3.35}$$

where

$$f_i = k(r_i - l_i) + b(\dot{r}_i - \dot{l}_i), \qquad i = 1, 2. \tag{3.36}$$

During single support, the swing leg is not in contact with the ground thus it cannot apply forces and the corresponding force f_i in Equation 3.35 goes to zero. The same is true for flight phase where both forces go to zero and the point mass simply follows a ballistic trajectory dictated by gravity.

3.2.4.2 Minimum Cost of Transport Problem

A common performance measure for locomotion systems is the mechanical cost of transport (COT). This dimensionless quantity is essentially a measure of the absolute work W required to travel some distance d scaled relative to mass m. The lower the number, the more efficient the system. We thus define our objective as

$$J = \frac{|\dot{r}_1 f_1| + |\dot{r}_2 f_2|}{mgd}, \tag{3.37}$$

3.2.4.3 Trajectory Optimization

In order to demonstrate the basics of phase scheduling, we pose the simple case of finding walking and running limit cycles for the target speed s. Knowing the phase sequence of running (single support $^{(ss)}$, flight $^{(f)}$) and walking (single support $^{(ss)}$, double support $^{(ds)}$), we can schedule the phases ahead of time. Because the springs are massless and there is no resulting impact, the phases can be linked by equating the boundary conditions.

$$\mathbf{q}^{(ss)}(t_f^{(ss)}) = \mathbf{q}^{(ds)}(t_0^{(ds)}) \tag{3.38}$$

$$\mathbf{q}^{(ds)}(t_f^{(ds)}) = \mathbf{q}^{(ss)}(t_0^{(ss)}) \tag{3.39}$$

where $\mathbf{q} = [r_i, \dot{r}_i, l_i, \dot{l}_i, \theta_i, \dot{\theta}_i]^\top$ for $i = 1, 2$.

We must now formulate a variation of our objective to eliminate the absolute value operator and the resulting non-smoothness. We can do this by defining a slack variable and applying the linear programming trick described by

$$min(|\dot{r}_i f_i|) \tag{3.40}$$

is equivalent to

$$min(W) \tag{3.41}$$

$$W \geq \dot{r}_i f_i \tag{3.42}$$

$$W \geq -\dot{r}_i f_i \tag{3.43}$$

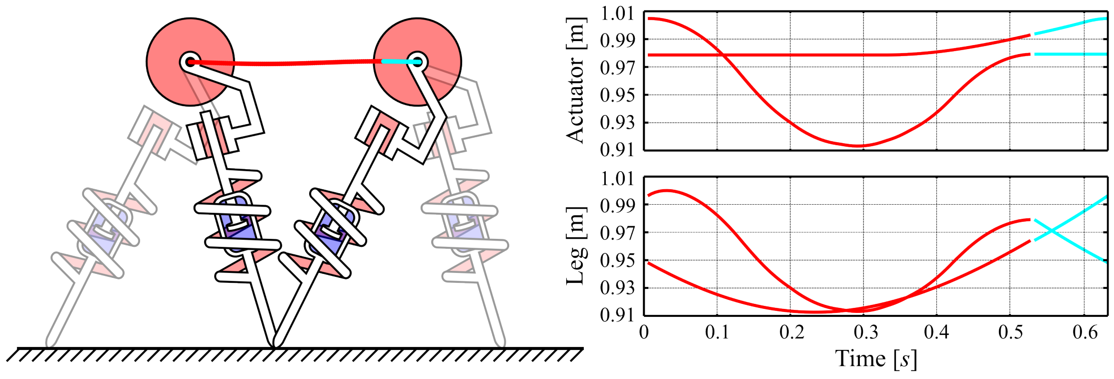

Figure 3.17: Illustration of the resulting cost-of-transport optimal walking gait for a non-ideal actuated spring-mass model. The resulting optimal control strategy re-injects lost energy into the system by pushing off with the trailing stance leg.

To constrain the gait speed, we constrain d to equal the difference in final and initial positions, and s as the total distance over time. Finally, we apply physical limitation constraints to the model of the form

$$0 \leq r_i \leq 1, \qquad z \geq 0, \qquad 0 \leq \theta_i \leq \pi, \qquad |\ddot{r}| \leq 1. \tag{3.44}$$

The resulting optimal limit cycle gaits are depicted in Figure 3.17. Using such a simple model allows us to intuitively investigate the found optimal strategies for fundamental insights that could be applicable to more complex systems. In this case, it is apparent that the controller prefers an extended push off of the trailing stance leg to add back any dissipated energy. This makes sense intuitively, as you would want to add energy back into the system when it is the least costly.

Chapter 4: Implementation

ATRIAS (**A**ssume **T**he **R**obot **I**s **A** **S**phere) is designed to be agile and efficient, which is accomplished by embracing underactuation. Specifically, we built the robot to approximate a simple, yet underactuated, dynamic model, the Spring-Loaded Inverted Pendulum. Among the primary manifestations of this design approach are lightweight legs to reduce impact losses and series springs to store and reinsert energy during dynamic maneuvers. A more detailed description of the design philosophy is presented in [22].

For our purposes, ATRIAS serves as a fitting testbed for our trajectory optimization and stabilization techniques. The springs and point feet all contribute to a robot with many degrees of underactuation, even when operating in just the sagittal plane. The robot is also designed such that actuators have a very indirect effect on the state of the system. Motors must first accelerate high-inertia rotors to deflect the in-series springs, and the springs must then apply forces through their own dynamics before any of the actuators' energy is imparted to the system. These slow dynamics between the motors and the gait mean that planning is likely very important to achieve locomotion. Lastly, gait economy is a driving goal of ATRIAS' construction, making it a well-suited target energy-optimal control.

4.1 System Model

Consider the planar model of ATRIAS illustrated in Figure 4.1. Due to the four bar mechanism in each leg, the configuration of the robot can be uniquely described using the generalized coordinates $[\theta_t, \theta_{1s}, \theta_{2s}, \theta_{1ns}, \theta_{2ns}]^{\mathsf{T}}$. We use the notation ns for non-stance leg and s for stance leg. Because the leg actuators are connected to the upper segments through series springs, additional coordinates $[\theta_{m1s}, \theta_{m2s}, \theta_{m1ns}, \theta_{m2ns}]^{\mathsf{T}}$ are used to model actuator dynamics. The complete generalize coordinates are therefore expressed by

$$\boldsymbol{q} = [\theta_t, \theta_{1s}, \theta_{2s}, \theta_{1ns}, \theta_{2ns}, \theta_{m1s}, \theta_{m2s}, \theta_{m1ns}, \theta_{m2ns}]^{\mathsf{T}} \tag{4.1}$$

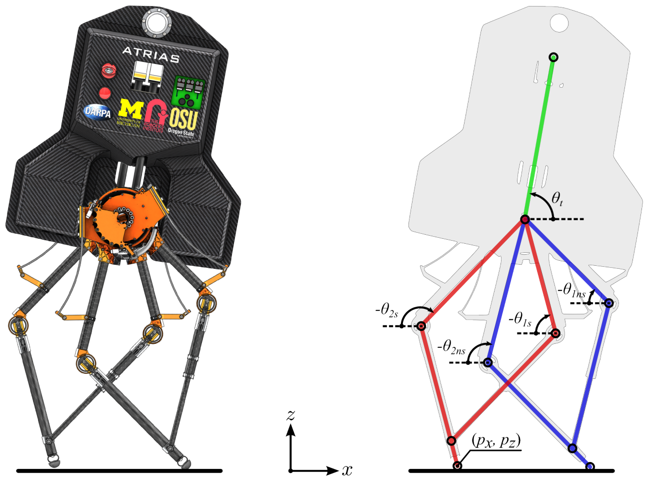

Figure 4.1: Illustration of the ATRIAS system model and generalized coordinates. The unique four-bar mechanisms on each leg minimizes leg mass and thus impact losses. The motors are located at the hips and are isolated from the load by series springs enabling highly dynamic maneuvers such as hopping.

The control inputs are the torques acting on the actuator rotors $\boldsymbol{u} = [\tau_{m1s}, \tau_{m2s}, \tau_{m1ns}, \tau_{m2ns}]^{\mathsf{T}}$. Even in the planar case the underactuated nature of ATRIAS is now directly apparent. With 9 generalized state coordinates and only 4 control inputs, ATRIAS is highly underactuated.

4.1.1 Single Support Model

In single support, the stance leg is pinned to the ground while the non-stance leg remains free. The model for the resulting open kinematic chain can be derived using the method of Lagrange. First the forward kinematics are derived and then the kinetic energy and potential energy of each link are computed. Summing the terms to get the total kinetic energy $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and total potential energy $V(\boldsymbol{q})$, we write the Lagrangian as

$$\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = T(\boldsymbol{q}, \dot{\boldsymbol{q}}) - V(\boldsymbol{q}) \tag{4.2}$$

Solving using the method of Lagrange and putting into general robot manipulator form results in

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} \tag{4.3}$$

where $\boldsymbol{H}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the Coriolis matrix, $\boldsymbol{G}(\boldsymbol{q})$ is the gravity vector, and $\boldsymbol{B}(\boldsymbol{q})$ is the mapping from joint torques to generalized forces.

4.1.2 Impact Model

The lower leg assemblies are sufficiently lightweight that we do not include impact losses resulting from swing leg touchdown. However, there is still an instantaneous change in velocities of the four-bar mechanism as the toe touches down. There is also a relabeling of coordinates as the non-stance leg becomes the new stance leg. This post-impact jump in states can be computed from the pre-impact states using the jump map defined by

$$\boldsymbol{x}^+ = \boldsymbol{\Delta}(\boldsymbol{x}^-) \tag{4.4}$$

where $\boldsymbol{x}^+ = [\boldsymbol{q}^+, \dot{\boldsymbol{q}}^+]$ is the state just after impact, $\boldsymbol{x}^- = [\boldsymbol{q}^-, \dot{\boldsymbol{q}}^-]$ is the state just before impact, and $\boldsymbol{\Delta}(\boldsymbol{x}^-)$ is the jump map.

4.2 System Identification

ATRIAS was painstakingly modeled with all components and material properties in SolidWorks to ensure parameters could be accurately estimated. Using this, the bulk of the system parameters were extracted leaving only spring and friction constants to be identified.

4.3 Trajectory Optimization

This section summarizes the many formulation details required to find an optimal trajectory for a full-order hybrid robot model.

Parameter	Description	Value
m_t	Torso Mass$^{(*)}$	$45.588\,[kg]$
r_t	Torso COM Location$^{(*)}$	$0.18116\,[m]$
G_a	Actuator Gear Ratio	50
I_a	Actuator Reflected Inertia$^{(*)}$	$3.0517\,[kg\,m^2]$
m_a	Actuator Mass$^{(*)}$	$2.8755\,[kg]$
b_a	Actuator Friction	$19\,[N\,m\,s/rad]$
k_τ	Actuator Torque Constant$^{(*)}$	$0.0987\,[N\,m\,/amp]$
k_s	Leaf Spring Constant	$1600\,[N\,m/rad]$
b_s	Leaf Spring Damping	$1.46\,[N\,m\,s/rad]$
l_1, l_2	Leg Segment Lengths$^{(*)}$	$0.5\,[m]$
k_{fs}	Static Coefficient of Friction$^{(*)}$	1.16

Table 4.1: List of ATRIAS parameters used for model simulation. Parameters marked with an asterisk are computed from specification sheets and SolidWorks model.

4.3.1 Minimum Cost of Transport

Similar to the reduced order template problem, we seek to minimize the cost of transport. This will not only attempt to minimize absolute mechanical work but maximize the distance traveled with each step. Cost of transport is also commonly used to benchmark and compare robot and animal gait performance. This makes it an ideal objective to minimize and thus our performance index is defined as

$$J = \frac{|\tau_{m1s}\dot{\theta}_{m1s}| + |\tau_{m2s}\dot{\theta}_{m2s}| + |\tau_{m1ns}\dot{\theta}_{m1ns}| + |\tau_{m2ns}\dot{\theta}_{m2ns}|}{(m_t + 4m_a)\,g\,d}, \tag{4.5}$$

Because the performance index must be smooth for convergence, the non-smooth absolute value operator must be eliminated. this is accomplished by defining a slack variable for each work term and applying the linear programming trick described by

$$min(|\dot{q}_i \tau_i|) \tag{4.6}$$

is equivalent to

$$min(W) \tag{4.7}$$

$$W \geq \dot{q}_i \tau_i \tag{4.8}$$

$$W \geq -\dot{q}_i \tau_i \tag{4.9}$$

4.3.2 Dynamic Constraints

As the number of degrees of freedom increases, the size of resulting dynamic differential equations also increases. This can lead to convergence issues during optimization, especially as the number of collocation nodes increases. To overcome this and to improve solver performance we take the idea of slack variables a step further. Typically, accelerations are explicitly solved for and the resulting equations are used directly in the numerical collocation constraints.

$$\ddot{\boldsymbol{q}} = \boldsymbol{H}(\boldsymbol{q})^{-1}(\boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})) \tag{4.10}$$

When done symbolically as COALESCE is designed to do, this results in enormous constraint equations that take substantially more time to evaluate. This is especially true for higher-order integration methods such as Hermite-Simpson and classical Runge-Kutta. Realizing and taking advantage of modern SQP algorithms abilities to solve systems of equations, we can leave the manipulator equation in standard form and simply create new acceleration slack variables for each state. The new slack variables $\ddot{q}_i$ are used in the collocation integration scheme but are additionally constrained to make dynamic sense by equating

$$0 = \boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} - (\boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})). \tag{4.11}$$

This formulation was tested on the triple, quadruple, and quintuple pendulum on a cart problems and resulted in the same solutions but in an order of magnitude less time than the standard formulation. Additionally, this slack dynamics formulation enabled solutions to be found for the decuple (ten-link), viguple (twenty-link), and triguple (thirty-link) pendulum on a cart swing-up problems.

4.3.3 Hybrid Transition Constraints

The impact model and limit cycle constraints implementation is fairly straight forward. The boundary conditions of a single single-support phase are constrained using the impact and relabeling model derived in Section 4.1.2. This ensures that only periodic orbits are considered.

4.3.4 Physical Constraints

ATRIAS has a number of physical limitations that could shape or exclude some otherwise desirable gait trajectories. It is therefore essential that we include additional physical limitation constraints to make sure the found gait is within the capabilities of the robot. Most obvious, is the actuator torque and velocity limits given by

$$-7.88 \leq [\dot{\theta}_{m1s}, \dot{\theta}_{m2s}, \dot{\theta}_{m1ns}, \dot{\theta}_{m2ns}] \leq 7.88, \tag{4.12}$$

$$-296.1 \leq [\tau_{m1s}, \tau_{m2s}, \tau_{m1ns}, \tau_{m2ns}] \leq 296.1 \tag{4.13}$$

Additionally, the four-bar mechanisms have built in hard-stops to limit range of motion (Figure 4.2. It is important that we stay away from these potentially dangerous areas to avoid damage. We therefore give a little extra buffer so the generated gaits don't push the limits to close. The resulting leg extension and flexion constraints are given by

$$0.60214 \leq [\theta_{2s} - \theta_{1s}, \theta_{2ns} - \theta_{1ns}] \leq 1.7453 \tag{4.14}$$

and the rotation relative to the torso is limited by

$$1.2654 \leq [\theta_t - \theta_{1s}, \theta_t - \theta_{1ns}] \leq 3.927 \tag{4.15}$$

$$2.3562 \leq [\theta_t - \theta_{2s}, \theta_t - \theta_{2ns}] \leq 5.0178 \tag{4.16}$$

Finally, because the pivot point at the toe is not actually a pin joint, we need to consider the force directions and cone of friction. To ensure that only positive vertical forces can be applied to the ground we limit

$$0 \leq [f_{z1}, f_{z2}] \leq \infty. \tag{4.17}$$

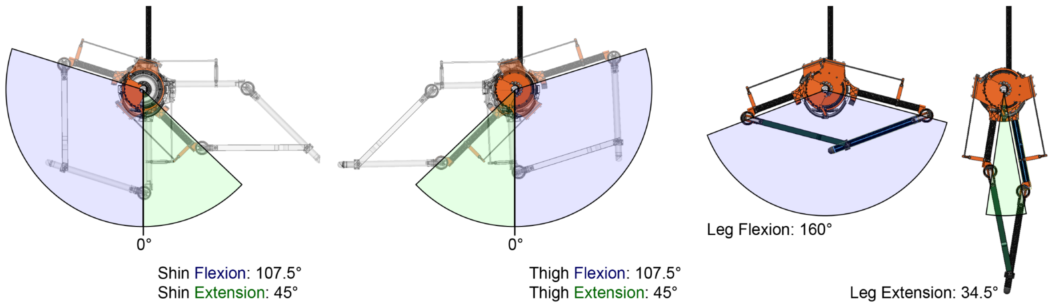

Figure 4.2: Illustration of the range of motion available in the ATRIAS four-bar leg mechanism. The leg has hard-stops that limit extension, flexion and swing of each coordinate.

To prevent the foot from applying large horizontal forces and slipping, we limit the forces to remain within the static cone of friction defined by

$$-\infty \leq [f_{x1} - k_{fs} f_{z1}, f_{x2} - k_{fs} f_{z2}] \leq 0, \tag{4.18}$$

$$0 \leq [f_{x1} + k_{fs} f_{z1}, f_{x2} - k_{fs} f_{z2}] \leq \infty. \tag{4.19}$$

4.3.5 Gait Constraints

There are a couple minor additions to the formulation that can help us shape or select the preferred gait. First we can set a target gait speed by constraining the horizontal distance d traveled to some specified value. For the illustrated examples we use a target speed of $0.75 m/s$. To prevent scuffing of the swing leg on the ground we set a foot clearance profile that the toe must stay above. This profile is defined using a sine wave and cross zero (the ground) at the moment of touch-down and take-off. The height of the apex can be adjusted depending on the desired clearance, to start we define a clearance of $3.0 cm$.

4.3.6 Result

A stroboscopic sequence of the resulting gait is illustrated in Figure 4.3. With as many state and control variables as ATRIAS has it is hard to make heads or tails of the corresponding time histories in any intuitive manner. It is much easier to look at the

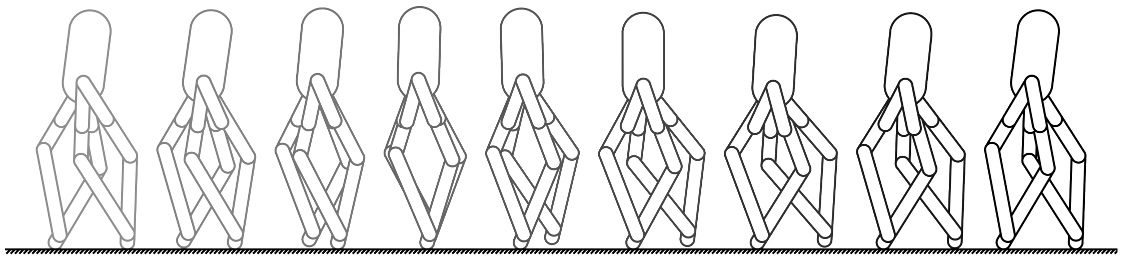
Figure 4.3: Stroboscopic sequence of the resulting cost-of-transport optimal walking gait. The controller appears to be exploiting the natural dynamics of the system, resulting in a very natural and bouncy gait.

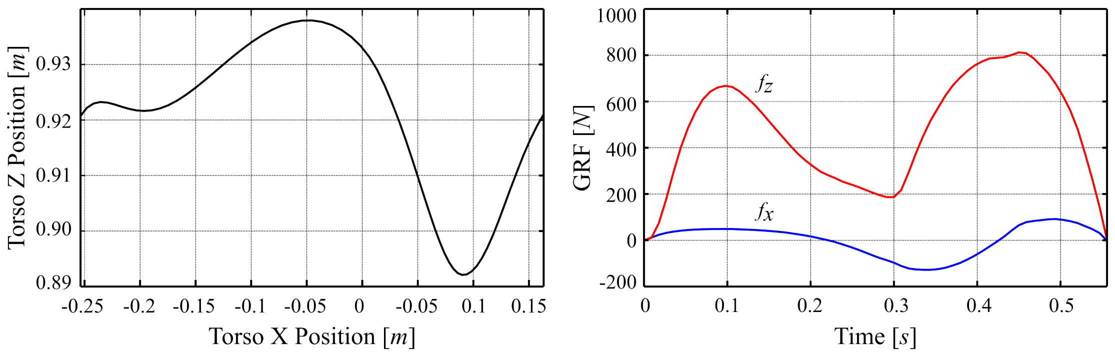

Figure 4.4: The ground reaction forces and center of mass trajectory depicted indicate that ATRIAS is indeed spring-mass like due to the common double hump vertical ground reaction forces.

global dynamics defined by the ground reaction forces and the center of mass trajectory. We can see from Figure 4.4, that the resulting vertical ground reaction forces have a nice double hump shape. This profile is commonly associated with human walking and spring-mass templates. It not surprising that ATRIAS exhibits this behavior as ATRIAS was designed to encapsulate the key dynamical features of these models.

4.4 Trajectory Stabilization

The resulting optimal trajectory is not open-loop stable and thus we need to derive a feedback controller to stabilize the trajectory. For this purpose we construct a time

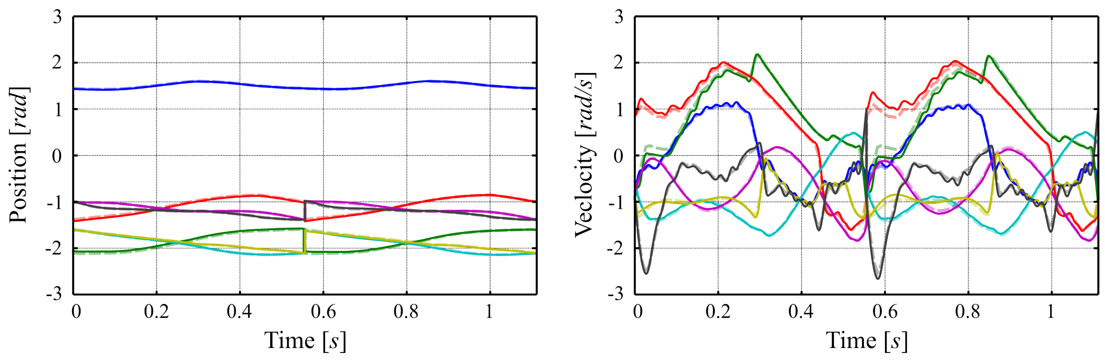

Figure 4.5: Stabilized walking gait on ATRIAS using trajectory optimization and LTV LQR. The dashed lines represent the optimal trajectories and the solid lines represent the stabilized trajectories.

varying linear quadratic regulator using the techniques described in Section 2.4.1. The key difference between the previously demonstrated implementations, is that now we now seek to stabilize a periodic orbit and not a finite horizon problem. This means we need to iterate our backward integration of the Riccati equation until the time varying solution $\boldsymbol{P}(t)$ converges. The resulting time varying gain schedule was turned into a controller and applied to a full-order model simulation. The resulting controller is able to stabilize the otherwise unstable trajectory and can recover from small initial condition perturbations. Figure 4.5 shows a two step recovery motion from a disturbance, demonstrating the effectiveness of the controller.

Chapter 5: Conclusion

This thesis set out to efficiently control a simulation of ATRIAS, a highly dynamic and underactuated bipedal robot. This task was approached as an optimal control problem, finding energy-optimal trajectories using numerical methods and stabilizing them with locally linear controllers. The resulting gait was stable and exhibited dynamical trademarks of ATRIAS' intended template, the spring-mass locomotion model. Particularly, the resulting gait bears the double-humped ground-reaction force characteristic of both spring-mass-model walking and human walking. We argue that this similarity to our target model is an indicator that the optimization is succeeding in maximally utilizing the dynamics of the machine.

In order to formulate and solve optimal control problems in an efficient manner, a framework entitled COALESCE was designed and built. This framework was coded in a high level language using an object oriented structure to make it user friendly and highly adaptable. It handles the direct transcription process, sparse symbolic Jacobian derivation and interfaces with many common optimization solvers enabling none expert users to quickly get started formulating problems. This framework was demonstrated on a variety of toy problems of increasing complexity and underactuation. A stabilization technique known as LTV LQR (Linear Time Varying Linear Quadratic Regulator) was used to demonstrate stabilization of these optimal control trajectories, a necessary step to bring feedback into the controller. This succession of increasingly complex examples culminated in a stabilized walking simulation of ATRIAS.

A few key take-away observations were found during this process. First, our solutions are not proven to be global, but there are many indicators that suggest they may be global. At the very least, these solutions aren't improvable in any obvious way, and certainly are not a product of numerical pseudo-minima. Due to COALESCE's embrace of symbolic formulations, high degree of freedom problems can become impractically immense when explicitly inverting the inertia matrix to solve for accelerations. We do have promising avenues to ameliorate this issue, and have preliminarily implemented a strategy using slack variables, leaving the equations of motion in manipulator form. This formulation has improved both build and solve performance by an order of magnitude and has been demonstrated on problems as large as the triguple (thirty-link) pendulum on a cart swing up maneuver thus far.

Bibliography

[1] Johan Åkesson. Optimicaan extension of modelica supporting dynamic optimization. In *Proc. 6th International Modelica Conference 2008*, 2008.

[2] MJ Anderson and WJ Grantham. Lyapunov optimal feedback control of a nonlinear inverted pendulum. *Journal of dynamic systems, measurement, and control*, 111(4):554–558, 1989.

[3] Karl Johan Åström and Katsuhisa Furuta. Swinging up a pendulum by energy control. *Automatica*, 36(2):287–295, 2000.

[4] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. Siam, 2010.

[5] John T Betts and William P Huffman. Sparse optimal control software socs. *Mathematics and Engineering Analysis Technical Document MEA-LR-085, Boeing Information and Support Services, The Boeing Company, PO Box*, 3707:98124–2207, 1997.

[6] Pranav A Bhounsule, Jason Cortell, and Andy Ruina. Design and control of ranger: an energy-efficient, dynamic walking robot. In *Proc. CLAWAR*, pages 441–448, 2012.

[7] Reinhard Blickhan. The spring-mass model for running and hopping. *Journal of biomechanics*, 22(11):1217–1227, 1989.

[8] Reinhard Blickhan and RJ Full. Similarity in multilegged locomotion: bouncing like a monopode. *Journal of Comparative Physiology A*, 173(5):509–517, 1993.

[9] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control.* CRC Press, 1975.

[10] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. Knitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

[11] Michael J Coleman, Anindya Chatterjee, and Andy Ruina. Motions of a rimless spoked wheel: a simple three-dimensional system with impacts. *Dynamics and Stability of Systems*, 12(3):139–159, 1997.

[12] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[13] Isabelle Fantoni, Rogelio Lozano, Mark W Spong, et al. Energy based control of the pendubot. *IEEE Transactions on Automatic Control*, 45(4):725–729, 2000.

[14] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM journal on optimization*, 12(4):979–1006, 2002.

[15] Tobias Glück, Andreas Eder, and Andreas Kugi. Swing-up control of a triple pendulum on a cart with experimental validation. *Automatica*, 49(3):801–808, 2013.

[16] Ambarish Goswami, Bernard Espiau, and Ahmed Keramane. Limit cycles in a passive compass gait biped and passivity-mimicking control laws. *Autonomous Robots*, 4(3):273–286, 1997.

[17] Knut Graichen, Michael Treuer, and Michael Zeitz. Fast side-stepping of the triple inverted pendulum via constrained nonlinear feedforward control design. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 1096–1101. IEEE, 2005.

[18] Knut Graichen, Michael Treuer, and Michael Zeitz. Swing-up of the double pendulum on a cart by feedforward and feedback control with experimental validation. *Automatica*, 43(1):63–71, 2007.

[19] CR HARGRAVES, SW PARIS, and WG VLASES. Otis past, present and future((optimal trajectories by implicit simulation)). In *AIAA, Guidance, Navigation and Control Conference, Hilton Head Island, SC*, page 1992, 1992.

[20] Masato Hirose and Kenichi Ogawa. Honda humanoid robots development. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):11–19, 2007.

[21] Daan Hobbelen, Tomas de Boer, and Martijn Wisse. System overview of bipedal robots flame and tulip: Tailor-made for limit cycle walking. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2486–2491. IEEE, 2008.

[22] C. Hubicki, J. Grimes, M. Jones, D. Renjewski, A. Spröwitz, A. Abate, and J. Hurst. ATRIAS: Enabling agile biped locomotion with a template-driven approach to robot design. *In Submission*, 2014.

[23] Kenji Kaneko, Fumio Kanehiro, Mitsuharu Morisawa, Kazuhiko Akachi, Gou Miyamori, Atsushi Hayashi, and Noriyuki Kanehira. Humanoid robot hrp-4-humanoid robotics platform with lightweight and slim body. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4400–4407. IEEE, 2011.

[24] MATLAB. *version 8.0 (R2012a)*. The MathWorks Inc., Natick, Massachusetts, 2012b.

[25] Tad McGeer. Passive dynamic walking. *the international journal of robotics research*, 9(2):62–82, 1990.

[26] Shozo Mori, Hiroyoshi Nishihara, and Katsuhisa Furuta. Control of unstable mechanical system control of pendulum. *International journal of Control*, 23(5):673–692, 1976.

[27] Ill-Woo Park, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. Mechanical design of the humanoid robot platform, hubo. *Advanced Robotics*, 21(11):1305–1322, 2007.

[28] Michael A Patterson and Anil V Rao. Gpops- ii: A matlab software for solving multiple-phase optimal control problems using hp–adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, 39(3), 2013.

[29] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[30] J Rubı, A Rubio, and A Avello. Swing-up control problem for a self-erecting double inverted pendulum. *IEE Proceedings-Control Theory and Applications*, 149(2):169–175, 2002.

[31] Mark W Spong. The swing up control problem for the acrobot. *Control Systems, IEEE*, 15(1):49–55, 1995.

[32] Mark W Spong and Daniel J Block. The pendubot: A mechatronic system for control research and education. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 555–556. IEEE, 1995.

[33] Oskar von Stryk. Users guide for dircol, a direct collocation method for the numerical solution of optimal control problems. version 2.1. *Simulation, Systems Optimization and Robotics Group, Technical University of Darmstadt. http://www. sim. informatik.//tu-darmstadt. de/index/leftnav. html. en*, 1999.

[34] Miomir Vukobratović and Branislav Borovac. Zero-moment pointthirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004.

[35] A Wächter and L Biegler. Ipopt-an interior point optimizer, 2009.

[36] XIN Xin and Masahiro Kaneda. The swing up control for the acrobot based on energy control approach. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 3, pages 3261–3266. IEEE, 2002.