DIRECT OPTIMIZATION OF ROBOT PARAMETERIZATION FOR TRAJECTORY PERFORMANCE

BY

MATHEW S. HALM

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelors of Science in Electrical Engineering
in the College of Engineering of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Seth Hutchinson

ABSTRACT

Much effort within the field of robotics has been made to study and mimic the agility of biological flight. Emulation of bat flight is particularly difficult, as bats utilize numerous independent means of control of both their inertial and aerodynamic characteristics to complete a variety of complex maneuvers. In this thesis, we investigate the viability of enabling a reduced-DoF bat robot to synthesize one such maneuver, inverted perching, by simultaneously and directly optimizing both the configuration and the trajectory of the robot.

We begin with a minimal model of a flapping flight system. Noting that longitudinal inertial dynamics represent the dominant behavior for the perching of biological bats, we introduce a single additional degree of actuation: a mass that may be shifted along the longitudinal axis of our system. We use the Lagrangian method to derive the equations of motion for our model, and then construct an augmented system where design parameters, namely linkage masses, are decision variables that are constrained to a constant value. We then reduce our optimization problem to an instance of the Direct Collocation trajectory optimization method, and find the minimum-time perching robot and trajectory. Our final configuration is able to complete the perching maneuver on a similar timescale to biological bats, suggesting viability of the reduced-DoF configuration.

Keywords: robotics, robot motion, trajectory optimization, design automation, biomimetics

ACKNOWLEDGMENTS

Just 18 short months ago, I found myself with a new major and a passion for robotics, but without any semblance of a plan for my career trajectory. I could not have asked for a better advisor than Prof. Hutchinson to help me get my bearings. He has consistently presented brilliant insight into how to face many of the technical and professional challenges that I've faced.

I would also like to thank my father. I don't think there are many men who can endure a decade of driving their wife to and from hospitals, working quite literally night and day, and showing up to an endless stream of school events, all while maintaining an unwaveringly supportive environment for their children. He remains the gold standard against which I measure my own character.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

B2	BatBot
DC	Direct Current
DoF	Degree of Freedom
SLIP	Spring-Loaded Inverted Pendulum
SNOPT	Sparse Nonlinear Optimizer
UAV	Unmanned Aerial Vehicle

CHAPTER 1

INTRODUCTION

1.1 Motivation

As the product of millions of years of evolution, biological systems fly with maneuverability and efficiency that far eclipse the performance of man-made machines [1]. In pursuit of replicating biological performance, many robotics researchers have built systems that mimic biological structures and morphologies (so-called *biomimetics*). In particular, there has been recent investment in emulating the flight of bats for their particular athleticism [2, 3, 4].

Biological bats are remarkably complex creatures. *Rousettus aegyptiacus*, or the Egyptian fruit bat, is able to execute aggressive maneuvers through the manipulation of the 40 degrees of freedom of the many actuated and passive joints of its musculoskeletal system [5]. Furthermore, the flight mechanics of *R. aegyptiacus* are made more complex by the somewhat unusual distribution of weight throughout its body. Unlike many other flapping-wing systems in biology, bats exhibit particularly heavy wings, which allow them to utilize inertial dynamics for attitude control during some modes of flight [6].

Perfect mimicry of such a complicated morphology is intractable due to the mass and volume of current actuation hardware. The challenge in effectively emulating such a system therefore lies in designing a low-DoF approximation that captures as much of the performance of the full systems as possible. As the resulting design space is highly constrained both in terms of morphology and performance, many roboticists seek some notion of optimality as a criterion for some parameterization of their systems.

Some roboticists optimize parameterization to be as similar as possible to the kinematics of biological systems [7]. There has been success in using these methods to capture the majority of the movement in bat wings using only a handful of DoFs [3]. However, there has also been great success in

using novel, low-DoF actuation that mimics humans' abstract understanding of how biological systems work [8]. Particularly, noting that bats manipulate their inertial parameters to control their attitude, Syed et al. [4] have demonstrated that with only a single DoF of inertial manipulation (manifested as a shifting mass), a bat-scale fixed-wing UAV is capable of executing aggressive attitude maneuvers.

As such novel systems do away with direct mimicry of biological structure to some degree, it becomes necessary to use an optimization function that is fundamentally different than those used in [3] and [7]. In this thesis, we present a general method for constructing an optimization problem that directly optimizes the parameterization of robots for trajectory performance, and apply it to the case of bat perching.

1.2 Approach

The functional basis of our method relies on methods used for trajectory optimization, typically used to synthesize a set of inputs that allow a robot to complete a maneuver in an optimal manner. In Chapter 3, we describe and then extend direct collocation, a method of generating a locally optimal trajectory computationally that relies on nonlinear programming. We do so by extending the dynamical model of the robot to contain the parameterization as explicit, constant decision variables. We then use this augmented system to generate an instance of direct collocation for which solutions contain both the optimal parameterization and the associated optimal trajectory.

In Chapter 4, we then apply this algorithm to a simplistic model of bat perching. Recent work has concluded that bats can execute large pitch rotations utilizing only longitudinal inertial dynamics, and that such behavior is dominant for inverted perching [4, 6]. Using this information, we create an approximate model of these dynamics by modeling a bat as a kinematic chain of rigid bodies. We then introduce a single-DoF mass-shifting actuator similar to the one used in [4]. An extended derivation of the equations of motion is presented in Appendix A. We then use the method described in Chapter 3 to create a program that will find an optimal mass distribution between the the legs, wings, and shifter. Our final configuration is able to complete the perching maneuver on a similar timescale to biological bats,

suggesting viability of the reduced-DoF configuration.

CHAPTER 2

LITERATURE REVIEW

2.1 Novel Biomimetic Robots

Biomimetic robotics most commonly employ novel hardware that relies on human intuition into the underlying behavior of biological systems. One of the most widely studied forms of robotic motion is legged locomotion. A very common form of robot structure and control system design is based on the spring-loaded inverted pendulum (SLIP) model [9]. SLIP is a hybrid approach that in its most basic form models a legged robot as a point mass with a single massless leg connecting the robot to the ground, all confined to the sagittal plane. The leg is modeled as containing active actuation as well as passive springs, which may be realized in hardware as control inputs to hip/knee motors that match the behavior of real springs. Poulakakis and Grizzle [10] extend SLIP to model a monopedal hopping robot, and synthesize a controller. Sreenath et al. also extend SLIP to reflect the dynamics of MABEL, a bipedal robot, on which the authors achieve stable walking motion [11].

Flapping fight has also been studied extensively. Paranjape, Chung, and Kim [8] describe the creation of a bird-like UAV, that uses dihedral actuation in the wings (up and down movement) to synthesize control inputs that effect reliable perching maneuvers using a "pitch up" motion to decelerate the vehicle. Ramezani et al. [2] detail a significantly more complex bat-like morphology. The iteration of the robot described in this paper, BatBot (B2), implements both dihedral and mediolateral (wing folding and unfolding) actuation in the forelimbs/wings was well as dorsoventral actuation in the hindlimbs. The increased complexity allows B2 to simulate the complex cyclic behavior of bats in nominal flight. Stable flight of B2 has been achieved in spite of passively unstable aerodynamics.

2.2 Optimal Biomimicry

Some biomimetic robots are designed to replicate biological systems in an optimal manner. Some of such robots achieve effective mimicry with only a few DoFs through leveraging of synergistic behavior. Originally formulated by Bernstein [12], the concept of biological *synergies* is that the dominant DoFs in many cases of animal motion are actually created through the coordination of multiple joints rather than individual ones. By finding the dominant synergies in a biological system, one may find that by directly implementing only a few of these synergies as single joints results in the capability to synthesize the dominant characteristics of the original motion. This is the case for human hands; 80% of hand motion can be represented using only the first two principal synergies [13]. Brown and Asada [14] use this conclusion to drive a 17-DoF hand using only two DC motors. Riskin et al. [5] perform a similar analysis on bat wings, showing that the first two principal synergies represent 57% of the mobility of *R. aegyptiacus*. Hoff et al. [3] also leverage bat wing synergies in order to create 2-DoF wings for an updated configuration of B2. After constraining the design space to be compatible with the B2 trunk, the authors generate parameters, implement hardware, and synthesize a wingbeat cycle that behaves similarly to that of biological bats.

CHAPTER 3

OPTIMAL ROBOT PARAMETERIZATION

3.1 Trajectory Optimization

Within the disciplines of control theory and motion planning, an often posed problem is to design an optimal input for some dynamical system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ over a time interval $[t_0, t_f]$. Optimality for such an input trajectory is often defined in terms of a *running cost* $l(\boldsymbol{x}(t), \boldsymbol{u}(t))$ and *final cost* $J_f(\boldsymbol{x}(t_f))$ that are combined to form the *total cost* J and associated optimization problem as follows:

$$J(\boldsymbol{x}(t_0), \boldsymbol{u}(\cdot)) = \int_{t_0}^{t_f} l(\boldsymbol{x}(t), \boldsymbol{u}(t))dt + J_f(\boldsymbol{x}(t_f)) \tag{3.1}$$

$$\begin{aligned} \underset{\boldsymbol{u}(\cdot)}{\text{minimize}} \quad & J(\boldsymbol{x}(t_0), \boldsymbol{u}(\cdot)) \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad \forall t \in [t_0, t_f] \end{aligned} \tag{3.2}$$

If the only constraints on such a problem are that the dynamics hold, it is possible that the problem may be using optimal control theory using tools such as the Hamilton-Jacobi-Bellman equation and Pontryagin's minimum principle [15], though such methods become infeasible for systems with high-dimensional or complicated equations of motion. Furthermore, for practical systems, intractability can arise through the complexity of additional constraints, including configuration space obstacles, actuator limits, and locally-applicable dynamics approximations, which are often necessary to produce feasible trajectories. When analytical solutions are not accessible, *trajectory optimization* methods are often used instead. Trajectory optimization methods describe a set of computational methods that can synthesize optimal state and input trajectories that conform to nonlinear constraints on states and inputs. For this thesis, we consider such problems that have nonlinear,

time-invariant state and input constraints $\boldsymbol{d}(\boldsymbol{x}(t), \boldsymbol{u}(t))$:

$$\begin{aligned} \underset{\boldsymbol{u}(\cdot)}{\text{minimize}} \quad & J(\boldsymbol{x}(t_0), \boldsymbol{u}(\cdot))) \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad \forall t \in [t_0, t_f] \\ & \boldsymbol{d}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq \boldsymbol{0} \quad \forall t \in [t_0, t_f] \end{aligned} \tag{3.3}$$

Trajectory optimization methods are often classified as *direct* or *indirect*. Indirect methods exactly define conditions for optimality, and then formulate a numerically solvable relaxation. Direct methods, by contrast, operate by forming a direct approximation of the optimization problem posed in Equation 3.3 [16]. The first step in forming a trajectory optimization problem by one of these methods is to *transcribe*, or discretize, candidate trajectories into a sequence of states and inputs, $(\boldsymbol{x}_0, \boldsymbol{u}_0), \ldots, (\boldsymbol{x}_N, \boldsymbol{u}_N)$. This transforms the trajectory space into a subset of $\mathbb{R}^n$ for some $n \in \mathbb{Z}$, which allows one to synthesize a parameter optimization problem that can produce approximate trajectories. Additionally, formulating the problem in terms of a parameter optimization allows for strong and versatile nonlinear optimization methods and software to be used. The Drake MATLAB library [17], used in this thesis, formulates many trajectory optimization problems as instances of parameter optimization that use the SNOPT [18] nonlinear optimization software.

3.2 Direct Collocation

Direct collocation is a direct transcription trajectory optimization method that leverages collocation methods to synthesize dynamics constraints. Each sequential pair of states and inputs $((\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}), (\boldsymbol{x}_i, \boldsymbol{u}_i))$ defines the value and time derivative of the state trajectory at times t_{i-1} and t_i to be $(\boldsymbol{x}_{i-1}, \boldsymbol{f}(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}))$ and $(\boldsymbol{x}_i, \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i))$, respectively. We assume that the input trajectory on the interval $[t_{i-1}, t_i]$ is an affine function of time as follows:

$$\tilde{\boldsymbol{u}}_i(t) = \boldsymbol{u}_{t-1} + \frac{(t - t_{i-1})}{\Delta t_i}(\boldsymbol{u}_t - \boldsymbol{u}_{t-1}) \tag{3.4}$$

where $\Delta t_i = t_i - t_{i-1}$, such that $\tilde{\boldsymbol{u}}_i(t_i) = \tilde{\boldsymbol{u}}_{i+1}(t_i) = \boldsymbol{u}_i$ and $\tilde{\boldsymbol{u}}_{i-1}(t_{i-1}) = \tilde{\boldsymbol{u}}_i(t_{i-1}) = \boldsymbol{u}_{i-1}$. The key assumption of direct collocation as originally posed by Hargraves and Paris is that for a sufficiently small $\Delta t_i = t_i - t_{i-1}$, the

state trajectory between $\boldsymbol{x}_{i-1}$ and $\boldsymbol{x}_i$ is approximately cubic [19]. If this cubic segment is to approximately conform to the dynamics, it follows that the derivative of the segment at $\boldsymbol{x}_{i-1}$ and $\boldsymbol{x}_i$ must necessarily be $\boldsymbol{f}(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1})$ and $\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i)$, respectively. A single cubic segment, the so-called Hermite cubic spline, can satisfy this condition. We can calculate this spline by first noting that it's value and time derivative are of the form

$$\tilde{\boldsymbol{x}}_i(t) = \boldsymbol{a}_1 + \boldsymbol{a}_2 t + \boldsymbol{a}_3 t^2 + \boldsymbol{a}_4 t^3 \tag{3.5}$$

$$\dot{\tilde{\boldsymbol{x}}}_i(t) = \boldsymbol{a}_2 + 2\boldsymbol{a}_3 t + 3\boldsymbol{a}_4 t^2 \tag{3.6}$$

where $\boldsymbol{a}_k$ are constant vectors of coefficients. We can determine the value of these coefficients by imposing the endpoint constraints $\tilde{\boldsymbol{x}}_i(t_{i-1}) = \boldsymbol{x}_{i-1}$, $\tilde{\boldsymbol{x}}_i(t_i) = \boldsymbol{x}_i$, $\dot{\tilde{\boldsymbol{x}}}_i(t_{i-1}) = \boldsymbol{f}(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1})$, and $\dot{\tilde{\boldsymbol{x}}}_i(t_i) = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i)$.

It is important to note that this spline would exist for any choice of $\boldsymbol{x}_{i-1}$ and $\boldsymbol{x}_i$; we have not yet introduced any constrictions on this trajectory to ensure that $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ approximately holds for the duration of the interval. Otherwise stated, we want to impose a constraint on $((\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}), (\boldsymbol{x}_i, \boldsymbol{u}_i))$ such that the Hermite spline between these points is close to the trajectory that the state of the system would follow given input $\tilde{\boldsymbol{u}}_i$ and initial condition $\boldsymbol{x}_{i-1}$.

The strategy of Hargraves and Paris to synthesize this constraint is to ensure that at the midpoint of the interval $[t_{i-1}, t_i]$, the value and derivative of the spline $(\boldsymbol{c}_i, \dot{\boldsymbol{c}}_i) = (\tilde{\boldsymbol{x}}_i(\frac{t_{i-1}+t_i}{2}), \dot{\tilde{\boldsymbol{x}}}_i(\frac{t_{i-1}+t_i}{2}))$, so-called "knot points," conform to the dynamics. To constructing this constraint, first we note that given (3.4), the input at this point is $\tilde{\boldsymbol{u}}_i(\frac{t_{i-1}+t_i}{2}) = \frac{1}{2}(\boldsymbol{u}_{i-1} + \boldsymbol{u}_i)$. Next, after solving for the spline coefficients, (3.5) and (3.6) yield

$$\boldsymbol{c}_i = \frac{1}{2}(\boldsymbol{x}_{i-1} + \boldsymbol{x}_i) + \frac{\Delta t_i}{8}(\boldsymbol{f}(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}) - \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i)) \tag{3.7}$$

$$\dot{\boldsymbol{c}}_i = -\frac{3}{2\Delta t_i}(\boldsymbol{x}_{i-1} - \boldsymbol{x}_i) - \frac{1}{4}(\boldsymbol{f}(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}) + \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i)) \tag{3.8}$$

With equations for the state, state derivative, and input, we now can create an equality constraint $\boldsymbol{g}_i$ that we can add to our optimization program to

enforce the dynamics at the midpoint:

$$\boldsymbol{g}_i = \boldsymbol{f}(\boldsymbol{c}_i, \frac{1}{2}(\boldsymbol{u}_{i-1} + \boldsymbol{u}_i)) - \dot{\boldsymbol{c}}_i = \boldsymbol{0} \tag{3.9}$$

A graphical representation of this mechanism can be seen in Figure 3.1. The accuracy over a single time-step of this method is fourth-order in Δt_i [20], resulting in highly accurate conformation to dynamics for tractable numbers of collocation points for many systems.

With the dynamics constraint approximately satisfied, we must now define reasonable approximations to the cost function and J and additional constraints $\boldsymbol{d}$ from (3.3). For the running cost l from (3.1), Drake approximates the integral with the trapezoidal rule to create the following cost function:

$$J(\boldsymbol{x}(t_0), \boldsymbol{u}(\cdot)) \approx \sum_{i=1}^{N} \frac{l_{i-1} + l_i}{2} \Delta t_i + J_f(\boldsymbol{x}_N) \tag{3.10}$$

where $l_i = l(\boldsymbol{x}_i, \boldsymbol{u}_i)$. Additionally, we impose the constraints of $\boldsymbol{d}$ by simply imposing them on the state and input of each time step. This leaves us with our final nonlinear program:

$$\begin{aligned} \underset{(\boldsymbol{x}_0,\boldsymbol{u}_0),\ldots,(\boldsymbol{x}_N,\boldsymbol{u}_N)}{\text{minimize}} \quad & \sum_{i=1}^{N} \frac{l_{i-1} + l_i}{2} \Delta t_i + J_f(\boldsymbol{x}_N) \\ \text{subject to} \quad & \Delta t_i > 0, \forall i \in 1, \ldots, N \\ & \boldsymbol{g}_i = \boldsymbol{0}, \forall i \in 1, \ldots, N \\ & \boldsymbol{d}(\boldsymbol{x}_i, \boldsymbol{u}_i) \geq \boldsymbol{0}, \forall i \in 0, \ldots, N \end{aligned} \tag{3.11}$$

3.3 Design Optimization for Trajectory Performance

A manual design process for developing a robot that can perform trajectories described by direct collocation might be to (1) come up with an initial design for the robot; (2) use direct collocation to determine the robot's capability to complete some trajectory; and finally, (3) if the performance is unsatisfactory, vary design parameters (e.g. component masses and dimensions) of the robot and apply optimization again. A natural automation of this design process would be to extend the trajectory optimization program by adding the design

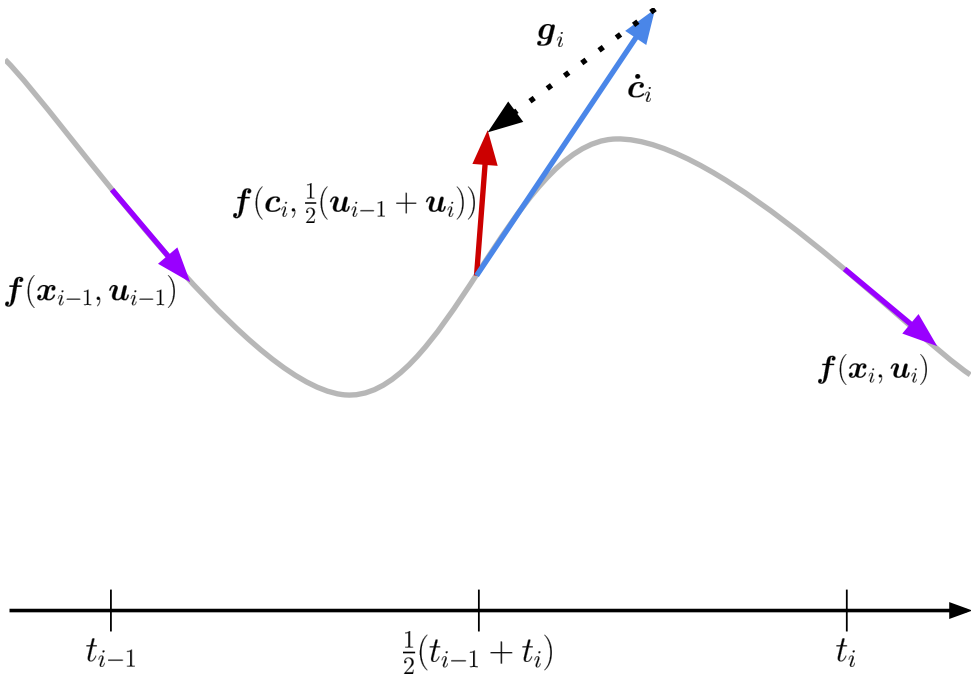

Figure 3.1: Visualization of $\boldsymbol{g}_i = \boldsymbol{0}$ constraint. Here we see the Hermite spline between the points $\boldsymbol{x}_{i-1}$ and $\boldsymbol{x}_i$, shown in gray. At the midpoint of the spline, we display the derivative of the spline, $\dot{\boldsymbol{c}}_i$, and display it in blue. We also display the what the derivative *would* be if the dynamics held in red. As the difference between these quantities $\boldsymbol{g}_i$ has nonzero magnitude, this particular spline is likely a poor representation of how the system would behave over this interval.

parameters to the set of optimization variables in the program. Specifically, instead of augmenting the optimization program formulation process, we chose to augment the *dynamical system*, such that our design optimization can reduced to a single instance of the original direct collocation algorithm.

We begin by noting for a dynamical model of a robot $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$, $\boldsymbol{f}$ may be viewed as an implicit function defined as

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{p}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}) \tag{3.12}$$

where $\boldsymbol{\alpha}$ is a constant vector of robot parameters as described above. We construct our augmented system by adding $\boldsymbol{\alpha}$ to our state vector, and enforcing constant value as follows:

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{\alpha}} \end{bmatrix} = \bar{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{0} \end{bmatrix} \tag{3.13}$$

We then construct the following generalization of the direct collocation

program in (3.11):

$$\begin{aligned} \underset{(\boldsymbol{y}_0,\boldsymbol{u}_0),\ldots,(\boldsymbol{y}_N,\boldsymbol{u}_N)}{\text{minimize}} \quad & \sum_{i=1}^{N} \frac{l_{i-1}+l_i}{2}\Delta t_i + J_f(\boldsymbol{x}_N) \\ \text{subject to} \quad & \Delta t_i > 0, \forall i \in 1,\ldots,N \\ & \bar{\boldsymbol{g}}_i = \boldsymbol{0}, \forall i \in 1,\ldots,N \\ & \boldsymbol{d}(\boldsymbol{x}_i,\boldsymbol{u}_i) \geq \boldsymbol{0}, \forall i \in 0,\ldots,N \end{aligned} \tag{3.14}$$

where $\bar{\boldsymbol{g}}_i$ are dynamical constraints constructed as follows according to (3.7) and (3.8):

$$\bar{\boldsymbol{c}}_i = \frac{1}{2}(\boldsymbol{y}_{i-1}+\boldsymbol{y}_i) + \frac{\Delta t_i}{8}(\bar{\boldsymbol{f}}(\boldsymbol{y}_{i-1},\boldsymbol{u}_{i-1}) - \bar{\boldsymbol{f}}(\boldsymbol{y}_i,\boldsymbol{u}_i)) \tag{3.15}$$

$$\dot{\bar{\boldsymbol{c}}}_i = -\frac{3}{2\Delta t_i}(\boldsymbol{y}_{i-1}-\boldsymbol{y}_i) - \frac{1}{4}(\bar{\boldsymbol{f}}(\boldsymbol{y}_{i-1},\boldsymbol{u}_{i-1}) + \bar{\boldsymbol{f}}(\boldsymbol{y}_i,\boldsymbol{u}_i)) \tag{3.16}$$

$$\bar{\boldsymbol{g}}_i = \bar{\boldsymbol{f}}(\bar{\boldsymbol{c}}_i, \tilde{\boldsymbol{u}}_i(\frac{t_{i-1}+t_i}{2})) - \dot{\bar{\boldsymbol{c}}}_i = 0 \tag{3.17}$$

The intent of this program is to obtain locally optimal parameters $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}_0^* = \ldots = \boldsymbol{\alpha}_N^*$ such that the plant $\dot{\boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{\alpha}^*}(\boldsymbol{x},\boldsymbol{u})$ can produce the most optimal trajectory as defined in (3.11), and that $(\boldsymbol{x}_0,\boldsymbol{u}_0),\ldots,(\boldsymbol{x}_N,\boldsymbol{u}_N)$ is the optimal trajectory for that system. First, we note that $\bar{\boldsymbol{c}}_i$ and $\dot{\bar{\boldsymbol{c}}}_i$ can be decomposed using (3.13) as follows:

$$\bar{\boldsymbol{c}}_i = \begin{bmatrix} \frac{1}{2}(\boldsymbol{x}_{i-1}+\boldsymbol{x}_i) + \frac{\Delta t_i}{8}(\boldsymbol{f}_{\boldsymbol{\alpha_{i-1}}}(\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1}) - \boldsymbol{f}_{\boldsymbol{\alpha_i}}(\boldsymbol{x}_i,\boldsymbol{u}_i)) \\ \frac{1}{2}(\boldsymbol{\alpha}_{i-1}+\boldsymbol{\alpha}_i) \end{bmatrix} \tag{3.18}$$

$$\dot{\bar{\boldsymbol{c}}}_i = \begin{bmatrix} -\frac{3}{2\Delta t_i}(\boldsymbol{x}_{i-1}-\boldsymbol{x}_i) - \frac{1}{4}(\boldsymbol{f}_{\boldsymbol{\alpha_{i-1}}}(\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1}) + \boldsymbol{f}_{\boldsymbol{\alpha_i}}(\boldsymbol{x}_i,\boldsymbol{u}_i)) \\ -\frac{3}{2\Delta t_i}(\boldsymbol{\alpha}_{i-1}-\boldsymbol{\alpha}_i) \end{bmatrix} \tag{3.19}$$

Given $\dot{\boldsymbol{\alpha}}(t) = 0$, (3.17) and (3.19) imply that $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_j \forall i,j \in 0,\ldots,N$ given that Δt_i are constrained to be positive by the program in (3.14). This implies that the exact effect of the dynamical constraints $\bar{\boldsymbol{g}}_i$ is to constrain the trajectory of $\boldsymbol{x}(t)$ precisely in the same manner as in (3.11) for a given constant $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0$. Therefore, given that they have identical cost functions,

the program in (3.14) is equivalent to (3.11), except that it additionally allows for manipulation of $\boldsymbol{\alpha}$ to further decrease the cost function.

CHAPTER 4

APPLICATION TO AERIAL ROBOTICS

4.1 Dynamical Model

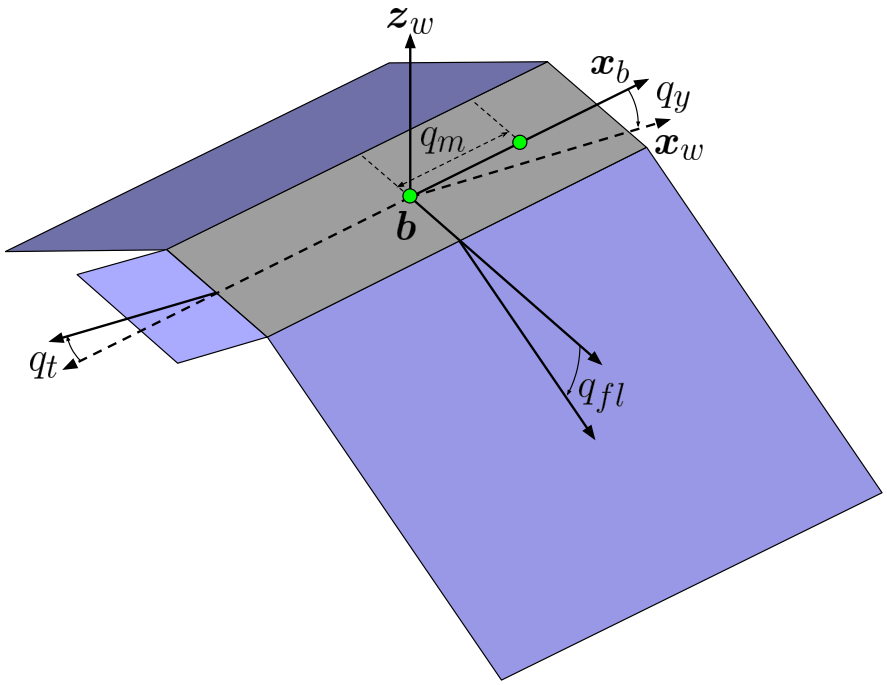

Figure 4.1: Kinematic diagram for bat system

For our minimalistic model of a perching bat, pictured in Figure 4.1, we constrict the trunk center of mass to the world xz-plane, and constrict roll and yaw angles to 0. We allow for three types of actuation: equal and opposite rotation of the wings about the longitudinal axis; equal rotation of the legs about the mediolateral axis; and mass-shifting along the longitudinal axis. Additionally, we assume that the robot consists only of five rigid bodies: the trunk, two wings, one leg structure, and the shifting mass. The remaining maneuvers can be adequately expressed with a dynamical system of the form

$$\boldsymbol{q} = \begin{bmatrix} q_y \\ b_x \\ b_z \\ q_{fl} \\ q_t \\ q_m \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}, \boldsymbol{u} = \begin{bmatrix} u_{fl} \\ u_t \\ u_m \end{bmatrix}, \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{4.1}$$

where q_y represents trunk rotation about the y-axis (pitch), b_x and b_z represent trunk position in the xz-plane, and $[q_{fl}\ q_t\ q_m]$ and $[u_{fl}\ u_t\ u_m]$ represent generalized coordinates and forces for wing, leg, and shifting mass joints respectively.

We derive $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ by using the Lagrangian method. We find the kinetic energy $T(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and potential energy $V(\boldsymbol{q})$ to be

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_i \frac{1}{2} m_i \dot{\boldsymbol{p}}_i^T \dot{\boldsymbol{p}}_i + \frac{1}{2} \boldsymbol{\omega}_i^T \boldsymbol{R}_i \boldsymbol{J}_i \boldsymbol{R}_i^T \boldsymbol{\omega}_i \tag{4.2}$$

$$V(\boldsymbol{q}) = \sum_i m_i g \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \boldsymbol{p}_i \tag{4.3}$$

where $\boldsymbol{p}_i$, $\dot{\boldsymbol{p}}_i$, $\boldsymbol{R}_i$, and $\boldsymbol{\omega}_i$, represent the position, linear velocity, orientation, and angular velocity of the ith body in the world frame; m_i and $\boldsymbol{J}_i$ represent the mass and principle-axes inertia tensor of the ith body; and g is the gravitational constant $9.81[\frac{m}{s^2}]$ [21]. These quantities can be read directly from q_y, b_x, b_z, and their derivatives for the trunk, and through the kinematic constraints arising from the joints, we can derive expressions for the rest of the bodies. As our model only captures inertial dynamics of a kinematic chain, we find the following expression for $\ddot{\boldsymbol{q}}$:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{B}\boldsymbol{u} \tag{4.4}$$

where $\boldsymbol{H}$ is the generalized mass-inertia matrix; $\boldsymbol{N}$ encompasses Coriolis, centrifugal, and gravity terms; and $\boldsymbol{B}$ is simply a constant $0-1$ matrix that maps each input to it's respective generalized force [22]. These matrices can be derived from T and V as follows:

$$\boldsymbol{H}(\boldsymbol{q}) = \frac{\partial^2 T}{\partial \dot{\boldsymbol{q}}^2} \tag{4.5}$$

$$\boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{\partial \boldsymbol{H}\dot{\boldsymbol{q}}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \frac{1}{2}\left(\frac{\partial \boldsymbol{H}\dot{\boldsymbol{q}}}{\partial \boldsymbol{q}}\right)^T \dot{\boldsymbol{q}} + \frac{\partial V}{\partial \boldsymbol{q}} \tag{4.6}$$

As $\boldsymbol{H}$ is well known to symmetric and positive definite [22], we can express $\boldsymbol{f}$ as

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H}(\boldsymbol{q})^{-1}(\boldsymbol{B}\boldsymbol{u} - \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}})) \end{bmatrix} \tag{4.7}$$

Expressions for $\boldsymbol{H}$, $\boldsymbol{N}$, and $\boldsymbol{B}$ as well as an extended derivation of the equations of motion are available in Appendix A.

4.2 Minimum-Time Perching

We now extend our model to formulate a direct design optimization problem to find masses for the non-trunk links (the wings, legs, and shifting mass) that will allow the robot to perform an inverted perching maneuver in minimum time. We assume that mass is uniformly distributed throughout the links, leaving 3 degrees of freedom in our design space: the total link masses $\boldsymbol{\alpha} = [\alpha_{fl}\ \alpha_t\ \alpha_m]^T$. This renders our extended dynamical system as

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{\alpha}} \end{bmatrix} = \bar{\boldsymbol{f}}(\boldsymbol{y},\boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{H}(\boldsymbol{q},\boldsymbol{\alpha})^{-1}(\boldsymbol{B}\boldsymbol{u} - \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{\alpha})) \\ \boldsymbol{0} \end{bmatrix} \tag{4.8}$$

via (3.13). Note that since T and V are dependent on the link masses, $\boldsymbol{H}$ and $\boldsymbol{N}$ are dependent on (and therefore functions of) $\boldsymbol{\alpha}$. We constrain the trajectory to a perching trajectory with the following boundary conditions:

$$\begin{bmatrix} q_y(0) \\ \dot{q}_y(0) \end{bmatrix} = \begin{bmatrix} -\frac{\pi}{3} \\ 0 \end{bmatrix}, \begin{bmatrix} q_y(t_f) \\ b_x(t_f) \\ b_z(t_f) \\ \dot{b}_x(t_f) \\ \dot{b}_y(t_f) \end{bmatrix} = \begin{bmatrix} -\pi \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.9}$$

which, simply put, constrains the beginning of the maneuver to a stable 60° pitch and the end of the maneuver to full inversion and zero trunk velocity at the perching location $(0, 0)$. We enforce bounding box constraints $\boldsymbol{r}_l < \boldsymbol{y}_i < \boldsymbol{r}_u$ over the entire trajectory to a) constrict the state to reasonable limits

to prevent unrealistic motion (e.g. wings clipping through the trunk) and b) constrain the link masses to feasible values. We also enforce symmetric actuator limitations $-\boldsymbol{s} < \boldsymbol{u}_i < \boldsymbol{s}$ that a) prevent motion that is much faster than reasonable for the number of time samples in the optimization program and b) prevent inputs that would be infeasible for typical linear actuators, servo motors, and brushless DS motors on the scales used for B2 and Allice [2] [4]. These limits are displayed in Tables 4.1 and 4.2. As we are constructing a minimum-time problem and explicitly constrain our final position, we are left with constant running and total costs $l(\boldsymbol{y}, \boldsymbol{u}) = 1$ and $J_f(\boldsymbol{y}_N) = 0$. Packing our constant constraints in (4.9) as $\boldsymbol{A}_0\boldsymbol{y}_0 = \boldsymbol{b}_0$ and$\boldsymbol{A}_N\boldsymbol{y}_N = \boldsymbol{b}_N$, we arrive to our final program formulation:

$$\begin{aligned} \underset{(\boldsymbol{y}_0,\boldsymbol{u}_0),\ldots,(\boldsymbol{y}_N,\boldsymbol{u}_N)}{\text{minimize}} \quad & \sum_{i=1}^{N} \Delta t_i \\ \text{subject to} \quad & \Delta t_i > 0, \forall i \in 1,\ldots,N \\ & \bar{\boldsymbol{g}}_i = \boldsymbol{0}, \forall i \in 1,\ldots,N \\ & \boldsymbol{r}_l < \boldsymbol{y}_i < \boldsymbol{r}_u, \forall i \in 0,\ldots,N \\ & -\boldsymbol{s} < \boldsymbol{u}_i < \boldsymbol{s}, \forall i \in 0,\ldots,N \\ & \boldsymbol{A}_0\boldsymbol{y}_0 = \boldsymbol{b}_0 \\ & \boldsymbol{A}_N\boldsymbol{y}_N = \boldsymbol{b}_N \end{aligned} \tag{4.10}$$

where N, the number of steps we chose, is 81. Numerical evaluation of this program will require constraint and cost gradients, which can easily be constructed given the Jacobians of $\bar{\boldsymbol{f}}$ with respect to $\boldsymbol{y}$ and $\boldsymbol{u}$, which are computed as

$$\frac{\partial \bar{\boldsymbol{f}}}{\partial \boldsymbol{y}} = \begin{bmatrix} \boldsymbol{0}_{6\times6} & \boldsymbol{I}_6 & \boldsymbol{0}_{6\times3} \\ \frac{\partial \boldsymbol{H}^{-1}}{\partial \boldsymbol{q}}\boldsymbol{V} + \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{N}}{\partial \boldsymbol{q}} & \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{N}}{\partial \dot{\boldsymbol{q}}} & \frac{\partial \boldsymbol{H}^{-1}}{\partial \boldsymbol{\alpha}}\boldsymbol{V} + \boldsymbol{H}^{-1}\frac{\partial \boldsymbol{N}}{\partial \boldsymbol{\alpha}} \\ \boldsymbol{0}_{3\times6} & \boldsymbol{0}_{3\times6} & \boldsymbol{0}_{3\times3} \end{bmatrix} \tag{4.11}$$

$$\frac{\partial \bar{\boldsymbol{f}}}{\partial \boldsymbol{u}} = \begin{bmatrix} \boldsymbol{0}_{6\times3} \\ \boldsymbol{H}^{-1}\boldsymbol{B} \\ \boldsymbol{0}_{3\times3} \end{bmatrix} \tag{4.12}$$

where $\boldsymbol{V} = \boldsymbol{B}\boldsymbol{u} - \boldsymbol{N}$ and $\frac{\partial \boldsymbol{H}^{-1}}{\partial \boldsymbol{q}} = -\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{q}}\boldsymbol{H}^{-1}$ is a third-order tenor rep-

Table 4.1: Continuous bounds on state trajectory

State	Units	Minimum	Maximum
q_y	rad	$-\infty$	$\frac{\pi}{10}$
b_x	m	-0.2	0.2
b_z	m	-50	0
q_{fl}	rad	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
q_t	rad	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
q_m	m	-0.04	0.04
$\dot{q}_y$	rad/s	$-\infty$	∞
$\dot{b}_x$	m/s	$-\infty$	∞
$\dot{b}_z$	m/s	$-\infty$	∞
$\dot{q}_{fl}$	rad/s	$-\infty$	∞
$\dot{q}_t$	rad/s	$-\infty$	∞
$\dot{q}_m$	m/s	$-\infty$	∞
α_{fl}	g	10	40
α_t	g	2.5	10
α_m	g	10	30

Table 4.2: Continuous bounds on input trajectory

Input	Units	Minimum	Maximum
u_{fl}	N·m	-0.036	0.036
u_t	N·m	-0.0002	0.0002
u_m	N	-0.021	0.021

resentation of the partial derivative of $\boldsymbol{H}^{-1}$ with respect to $\boldsymbol{q}$ (and similarly with $\frac{\partial \boldsymbol{H}^{-1}}{\partial \boldsymbol{\alpha}}$).

4.3 Results

Our program output a minimum-time perching trajectory of length $t = 266$ [ms]. A visualization of this motion can be found in Figure 4.2, and a plot of the trunk trajectory is displayed in Figure 4.3. As shown in Table 4.3, all of the generated optimal mass parameters were found to be constant within an extremely small tolerance, and centered at an extreme of the allowable range.

It is worth noting that two of these parameters, both α_t and α_m, are at the maximum of their allowable range. This is not entirely unexpected; having a high mass allows for more angular momentum to be exerted on

Table 4.3: Optimal mass parameters for minimum-time perching

Parameter	Range [g]
α_{fl}	$10 \pm 4.09 \times 10^{-14}$
α_t	$10 \pm 1.82 \times 10^{-15}$
α_m	$30 \pm 1.7 \times 10^{-16}$

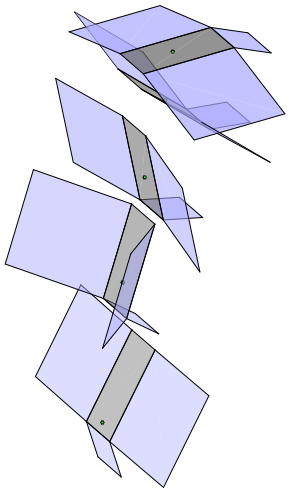

Figure 4.2: Visualization of perching trajectory

the trunk before hitting an actuator stroke limit. However, high mass is detrimental to steady-state flight performance, as it decreases the thrust to weight ratio. We would not expect aerodynamics to have affected our results, as they were not modeled, but it would likely be advisable to have some constraints on minimum aerodynamic performance before generating a final parameterization to guarantee that steady-state flight is, at the very least, feasible. We also note that if, similar to Allice [4], the primary weight of the mass-shifter is simply the battery, that having a heavy mass-shifter would be feasible without significantly affecting the total mass.

We also acknowledge that we have not guaranteed that the results of our program are insensitive to the minute fluctuations of $\boldsymbol{\alpha}$ over the trajectory caused by rounding error. In order to verify that this trajectory is feasible for exactly constant mass parameters, we take the timewise average of these parameters, create an instance of the system described in (4.7), and simulate

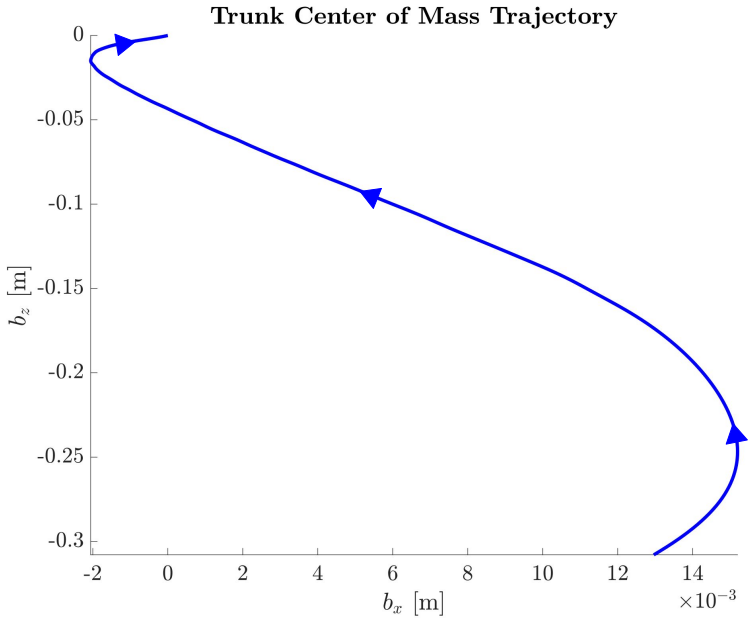

Figure 4.3: Trajectory of trunk center of mass in world xz-plane

the maneuver open-loop for a single bat using MATLAB's ode45 (which uses an explicit Runge-Kutta formulation [23]). We plot both the Runge-Kutta and collocation pitch trajectories in Figure 4.4. Additionally, we treat the Runge-Kutta results as ground truth for the trajectory capability of the robot, and plot the absolute value of relative error accumulation on the collocation pitch trajectory ($-q_y$) in Figure 4.5. It is clear from the graph that the Runge-Kutta and collocation results begin to diverge, but given that they stay within 1.2×10^{-5} [rad] of each other, we conclude that for such a short interval, the collocation trajectory is reasonably representative of the capabilities of the modeled dynamics. We also note that there appears to be high-frequency noise in the error graph. As the input is modeled as piecewise linear, its second derivative is unbounded in several locations. We hypothesize that truncation error associated with these points would be enough to cause the fluctuations in the figure.

We also observe in Figure 4.6 that the leg and shifting mass inputs were at extremal values for nearly the entire trajectory, suggesting that bang-bang control may be optimal for these inputs. However, the wing input signal has significant high frequency content, which could possibly be damped by adding an input component to l. We also note that due to the relatively high

mass of the wings, that this oscillatory behavior is significantly attenuated in the wing joint trajectory shown in Figure 4.7.

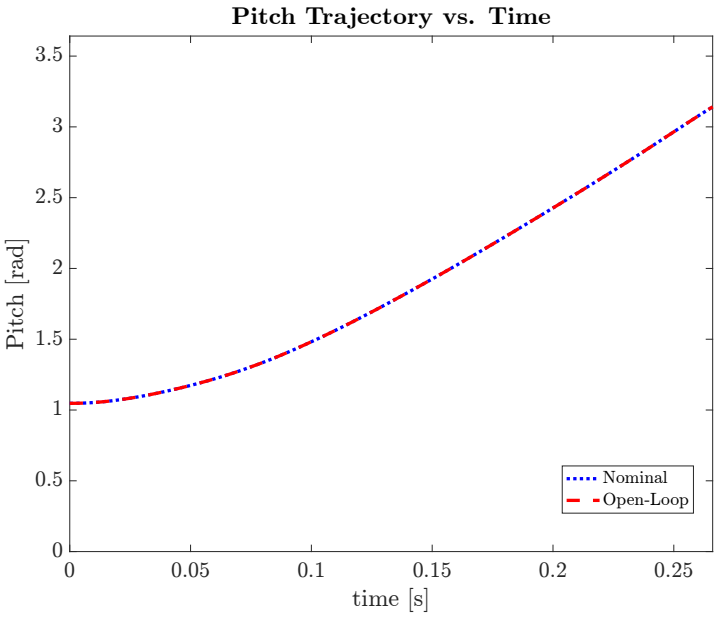

Figure 4.4: Pitch trajectory

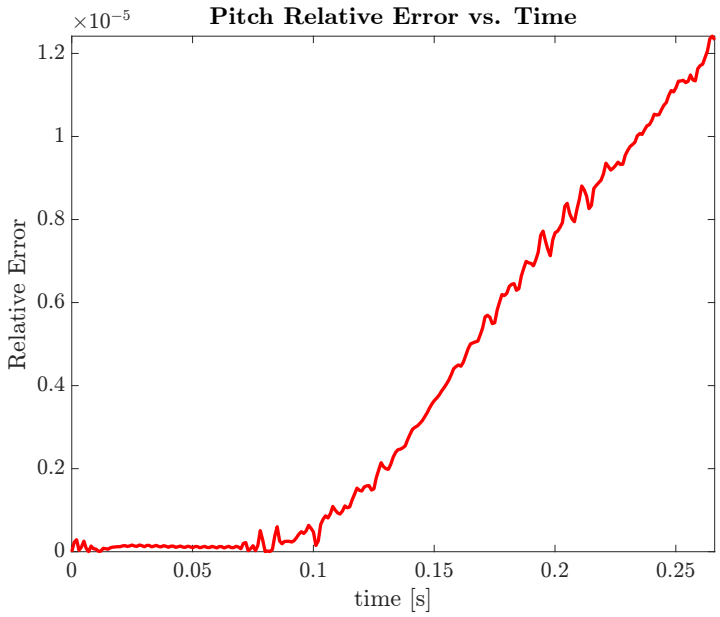

Figure 4.5: Open loop pitch relative error (absolute)

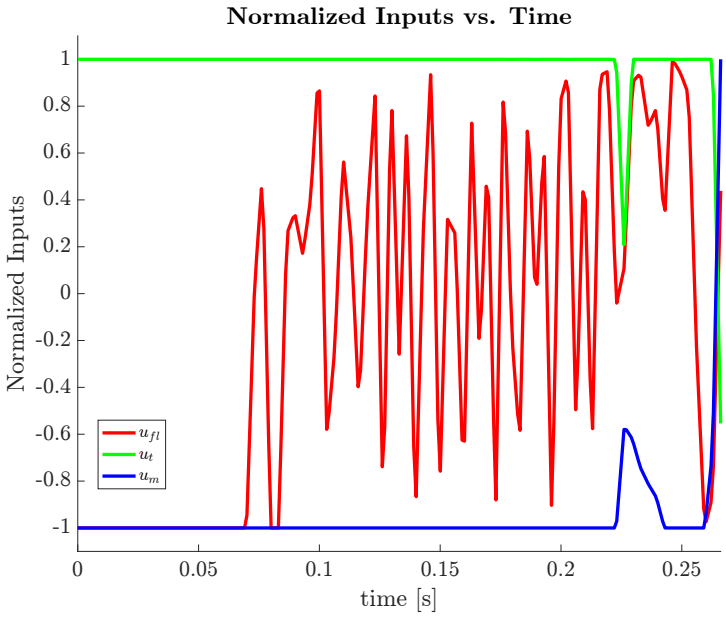

Figure 4.6: Input trajectories, with constraint range normalized to $[-1, 1]$

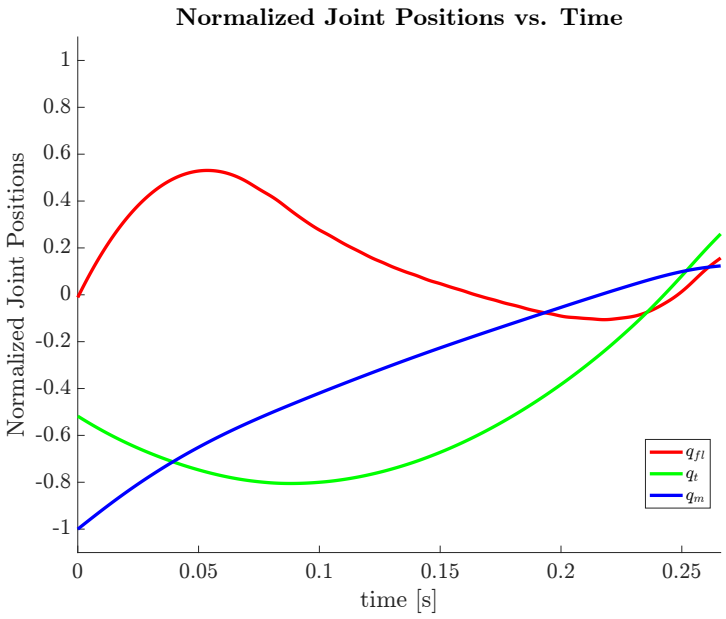

Figure 4.7: Joint trajectories, with constraint range normalized to $[-1, 1]$

CHAPTER 5

CONCLUSION

In this work, we have presented a general method for optimizing the parameterization of robot hardware for trajectory performance. Using an instance of Direct Collocation, we are able to simultaneously generate a set of optimal mass parameters and an optimal trajectory for minimum-time inverted perching of a simplistic model of a biomimetic, bat-like robot. In simulation, we achieved a trajectory duration of 266 [ms], which is comparable to that of a biological bat.

While the minimum-time perching problem was solved in this thesis, in practice, it may be wise to choose a different optimization function and model for a general-purpose robot, such as maximum efficiency steady-state flight. Subsequent work may examine the creation of such an optimization problem, which would require modeling of the aerodynamics of the robot.

APPENDIX A

EXTENDED DYNAMICS DERIVATIONS

From [21], we know that for a system of connected rigid bodies, we can express the total kinetic and potential energy as

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_i \frac{1}{2} m_i \dot{\boldsymbol{p}}_i^T \dot{\boldsymbol{p}}_i + \frac{1}{2} \boldsymbol{\omega}_i^T \boldsymbol{R}_i \boldsymbol{J}_i \boldsymbol{R}_i^T \boldsymbol{\omega}_i \tag{A.1}$$

$$V(\boldsymbol{q}) = \sum_i m_i g \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \boldsymbol{p}_i \tag{A.2}$$

where $\boldsymbol{p}_i$, $\dot{\boldsymbol{p}}_i$, $\boldsymbol{R}_i$, and $\boldsymbol{\omega}_i$, represent the position, linear velocity, orientation, and angular velocity of the ith body in the world frame; m_i and $\boldsymbol{J}_i$ represent the mass and principle-axes inertia tensor of the ith body; and g is the gravitational constant $9.81[\frac{m}{s^2}]$ [21]. The configuration $\boldsymbol{q}$ of the system by definition contains enough information to determine the world-frame location of each point on the robot, so each $\boldsymbol{p}_i$ can be written as a function of $\boldsymbol{q}$ [21]. We can use this formulation to derive the linear velocities as

$$\dot{\boldsymbol{p}}_i = \frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \tag{A.3}$$

Similarly, we can derive the body orientations $\boldsymbol{R}_i(\boldsymbol{q})$ from the configuration, and the angular velocities as

$$\boldsymbol{S}(\boldsymbol{\omega}_i) = \dot{\boldsymbol{R}}_i \boldsymbol{R}_i^T = \left(\sum_j \frac{\partial \boldsymbol{R}_i}{\partial q_j} \dot{q}_j \right) \boldsymbol{R}_i^T \tag{A.4}$$

where $\boldsymbol{S}$ transforms a vector into a skew symmetric matrix as follows:

$$\boldsymbol{S}\left(\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \right) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{A.5}$$

Spong et al. use this structure to rewrite the kinetic energy as

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{H}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{A.6}$$

Using the Lagrangian method, we can write the equations of motion as

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \boldsymbol{q}} = \boldsymbol{Q} \tag{A.7}$$

where $L = T - V$ and $\boldsymbol{Q}$ represents nonconservative generalized forces applied to the system. Substituting for T,

$$\begin{aligned}
\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \boldsymbol{q}} &= \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}} - \frac{\partial V}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial T}{\partial \boldsymbol{q}} + \frac{\partial V}{\partial \boldsymbol{q}} \\
&= \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial T}{\partial \boldsymbol{q}} + \frac{\partial V}{\partial \boldsymbol{q}} \\
&= \frac{d}{dt}(\boldsymbol{H}\dot{\boldsymbol{q}}) - \frac{\partial}{\partial \boldsymbol{q}}\frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{H}\dot{\boldsymbol{q}} + \frac{\partial V}{\partial \boldsymbol{q}} \\
&= \dot{\boldsymbol{H}}\dot{\boldsymbol{q}} + \boldsymbol{H}\ddot{\boldsymbol{q}} - \frac{\partial}{\partial \boldsymbol{q}}\frac{1}{2}(\boldsymbol{H}\dot{\boldsymbol{q}})^T\dot{\boldsymbol{q}} + \frac{\partial V}{\partial \boldsymbol{q}} \\
&= \boldsymbol{H}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{H}}\dot{\boldsymbol{q}} - \frac{1}{2}\left(\frac{\partial \boldsymbol{H}\dot{\boldsymbol{q}}}{\partial \boldsymbol{q}}\right)^T\dot{\boldsymbol{q}} + \frac{\partial V}{\partial \boldsymbol{q}}
\end{aligned} \tag{A.8}$$

Using $\boldsymbol{H}_k$ to denote the kth column of $\boldsymbol{H}$, we note that

$$
\begin{aligned}
\dot{\boldsymbol{H}}\dot{\boldsymbol{q}} &= \sum_i \dot{\boldsymbol{H}}_i \dot{q}_i \\
&= \sum_i \left(\sum_j \frac{\partial \boldsymbol{H}_i}{\partial q_j} \dot{q}_j \right) \dot{q}_i \\
&= \sum_i \sum_j \frac{\partial \boldsymbol{H}_i}{\partial q_j} \dot{q}_j \dot{q}_i \\
&= \sum_j \sum_i \frac{\partial \boldsymbol{H}_i}{\partial q_j} \dot{q}_i \dot{q}_j \\
&= \sum_j \sum_i \frac{\partial \boldsymbol{H}_i}{\partial q_j} \dot{q}_i \dot{q}_j \\
&= \sum_j \frac{\partial \boldsymbol{H}}{\partial q_j} \dot{\boldsymbol{q}} \dot{q}_j \\
&= \sum_j \frac{\partial \boldsymbol{H}\dot{\boldsymbol{q}}}{\partial q_j} \dot{q}_j \\
&= \frac{\partial \boldsymbol{H}\dot{\boldsymbol{q}}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}
\end{aligned}
\tag{A.9}
$$

Taking the inputs of the system to be generalized forces on the wing, tail, and shifting mass joints, we can formulate $\boldsymbol{Q}$ from (A.7) as a function of $\boldsymbol{u}$:

$$
\boldsymbol{Q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ u_{fl} \\ u_t \\ u_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_{3\times 3} \\ \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{u} = \boldsymbol{B}\boldsymbol{u} \tag{A.10}
$$

Combining (A.7)-(A.10), we are left with our final formulation of the equations of motion,

$$
\boldsymbol{H}\ddot{\boldsymbol{q}} + \frac{\partial \boldsymbol{H}\dot{\boldsymbol{q}}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} - \frac{1}{2} \left(\frac{\partial \boldsymbol{H}\dot{\boldsymbol{q}}}{\partial \boldsymbol{q}} \right)^T \dot{\boldsymbol{q}} + \frac{\partial V}{\partial \boldsymbol{q}} = \boldsymbol{B}\boldsymbol{u} \tag{A.11}
$$

which, given the definitions of $\boldsymbol{H}$ and $\boldsymbol{N}$ from (4.5) and (4.6) are equivalent to the form given in (4.4).

We now derive forms for $\boldsymbol{p}_i$, $\boldsymbol{R}_i$, and $\boldsymbol{J}_i$ for our bat robot given the following assumptions:

- Each link is a uniform-density rectangle or rectangular prism. The trunk is a prism with body-axis dimensions $\begin{bmatrix} b_{lx} & b_{ly} & b_{lz} \end{bmatrix}^T$ and mass m_b. The wings are square with side length b_{lx}. The tail structure is square with side length b_{ly}. The mass shifter is a cube with side length c_l.

- Each wing is attached to the trunk such that one side is coincident with a side of the trunk prism and aligned with the trunk's body x-axis. The joints connecting the left and right wing to the trunk are at $\pm \begin{bmatrix} 0 & \frac{1}{2}b_{ly} & 0 \end{bmatrix}^T$ in the trunk frame, respectively. Otherwise stated, the seams between the wings and trunk split the trunk's side faces lengthwise.

- The tail is attached to the end of the trunk such that one side is coincident with the trunk prism's rear face and aligned to the trunk's body y-axis. The joint connecting the tail to the trunk is located at $\begin{bmatrix} -\frac{1}{2}b_{lx} & 0 & 0 \end{bmatrix}^T$ in the trunk frame. Otherwise stated, the seam between the tail and trunk splits the rear face horizontally.

as well as the following comments on notation:

- $\boldsymbol{R}_x(\theta)$, $\boldsymbol{R}_y(\theta)$, and $\boldsymbol{R}_z(\theta)$ denote rotations by θ radians about the x, y, and z axes, respectively.

- For any k, c_k and s_k are shorthand for $\cos(q_k)$ and $\sin(q_k)$, respectively.

- The trunk, left wing, right wing, tail, and mass shifter will be numbered as bodies 1 through 5.

$$\boldsymbol{R}_1 = \boldsymbol{R}_y(q_y) \tag{A.12}$$
$$\boldsymbol{R}_2 = \boldsymbol{R}_1\boldsymbol{R}_x(q_{fl}) \tag{A.13}$$
$$\boldsymbol{R}_3 = \boldsymbol{R}_1\boldsymbol{R}_x(-q_{fl}) \tag{A.14}$$
$$\boldsymbol{R}_4 = \boldsymbol{R}_1\boldsymbol{R}_y(q_t) \tag{A.15}$$
$$\boldsymbol{R}_5 = \boldsymbol{R}_1 \tag{A.16}$$

$$\boldsymbol{p}_1 = \begin{bmatrix} b_x \\ 0 \\ b_z \end{bmatrix} \tag{A.17}$$

$$\boldsymbol{p}_2 = \boldsymbol{p}_1 + \boldsymbol{R}_1 \begin{bmatrix} 0 \\ \frac{1}{2}b_{ly} \\ 0 \end{bmatrix} + \boldsymbol{R}_2 \begin{bmatrix} 0 \\ \frac{1}{2}b_{lx} \\ 0 \end{bmatrix} \tag{A.18}$$

$$\boldsymbol{p}_3 = \boldsymbol{p}_1 + \boldsymbol{R}_1 \begin{bmatrix} 0 \\ -\frac{1}{2}b_{ly} \\ 0 \end{bmatrix} + \boldsymbol{R}_2 \begin{bmatrix} 0 \\ -\frac{1}{2}b_{lx} \\ 0 \end{bmatrix} \tag{A.19}$$

$$\boldsymbol{p}_4 = \boldsymbol{p}_1 + \boldsymbol{R}_1 \begin{bmatrix} -\frac{1}{2}b_{lx} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{R}_2 \begin{bmatrix} -\frac{1}{2}b_{ly} \\ 0 \\ 0 \end{bmatrix} \tag{A.20}$$

$$\boldsymbol{p}_5 = \boldsymbol{p}_1 + \boldsymbol{R}_1 \begin{bmatrix} q_m \\ 0 \\ 0 \end{bmatrix} \tag{A.21}$$

$$\boldsymbol{J}_1 = \frac{m_b}{12} \begin{bmatrix} b_{ly}^2 + b_{lz}^2 & 0 & 0 \\ 0 & b_{lx}^2 + b_{lz}^2 & 0 \\ 0 & 0 & b_{lx}^2 + b_{ly}^2 \end{bmatrix} \tag{A.22}$$

$$\boldsymbol{J}_2 = \frac{\alpha_{fl}}{12} \begin{bmatrix} b_{lx}^2 & 0 & 0 \\ 0 & b_{lx}^2 & 0 \\ 0 & 0 & 2b_{lx}^2 \end{bmatrix} \tag{A.23}$$

$$\boldsymbol{J}_2 = \frac{\alpha_{fl}}{12} \begin{bmatrix} b_{lx}^2 & 0 & 0 \\ 0 & b_{lx}^2 & 0 \\ 0 & 0 & 2b_{lx}^2 \end{bmatrix} \tag{A.24}$$

$$\boldsymbol{J}_4 = \frac{\alpha_t}{12} \begin{bmatrix} b_{ly}^2 & 0 & 0 \\ 0 & b_{ly}^2 & 0 \\ 0 & 0 & 2b_{ly}^2 \end{bmatrix} \tag{A.25}$$

$$\boldsymbol{J}_5 = \frac{\alpha_m}{6} c_l^2 \boldsymbol{I}_3 \tag{A.26}$$

Using (A.3) and (A.4), we can directly compute both potential and kinetic energy, which we can use to generate $\boldsymbol{H}$ and $\boldsymbol{N}$ via (4.5) and (4.6). The final

formulas as well are provided below. Table A.1 lists the numerical values of various static masses and dimensions.

$$
\begin{aligned}
\boldsymbol{H}_{1,1} &= \frac{{b_{lx}}^2 m_b}{12} + \frac{{b_{lz}}^2 m_b}{12} + \frac{{b_{lx}}^2 \alpha_t}{4} + \frac{{b_{ly}}^2 \alpha_t}{3} + \frac{{b_{lx}}^2 \alpha_{fl}}{6} + \frac{{c_l}^2 \alpha_m}{6} \\
&\quad + \alpha_m {q_c}^2 + \frac{2{b_{lx}}^2 \alpha_{fl} {s_{fl}}^2}{3} + \frac{b_{lx} b_{ly} \alpha_t c_t}{2} \\
\boldsymbol{H}_{1,2} &= \frac{b_{lx} \alpha_t s_y}{2} - \alpha_m q_c s_y + \frac{b_{ly} \alpha_t \sin(q_t + q_y)}{2} + b_{lx} \alpha_{fl} c_y s_{fl} \\
\boldsymbol{H}_{1,3} &= \frac{b_{lx} \alpha_t c_y}{2} - \alpha_m q_c c_y + \frac{b_{ly} \alpha_t \cos(q_t + q_y)}{2} - b_{lx} \alpha_{fl} s_{fl} s_y \\
\boldsymbol{H}_{1,4} &= 0 \\
\boldsymbol{H}_{1,5} &= \frac{b_{ly} \alpha_t (4 b_{ly} + 3 b_{lx} c_t)}{12} \\
\boldsymbol{H}_{1,6} &= 0 \\
\boldsymbol{H}_{2,1} &= \frac{b_{lx} \alpha_t s_y}{2} - \alpha_m q_c s_y + \frac{b_{ly} \alpha_t \sin(q_t + q_y)}{2} + b_{lx} \alpha_{fl} c_y s_{fl} \\
\boldsymbol{H}_{2,2} &= m_b + \alpha_m + \alpha_t + 2\alpha_{fl} \\
\boldsymbol{H}_{2,3} &= 0 \\
\boldsymbol{H}_{2,4} &= b_{lx} \alpha_{fl} c_{fl} s_y \\
\boldsymbol{H}_{2,5} &= \frac{b_{ly} \alpha_t \sin(q_t + q_y)}{2} \\
\boldsymbol{H}_{2,6} &= \alpha_m c_y \\
\boldsymbol{H}_{3,1} &= \frac{b_{lx} \alpha_t c_y}{2} - \alpha_m q_c c_y + \frac{b_{ly} \alpha_t \cos(q_t + q_y)}{2} - b_{lx} \alpha_{fl} s_{fl} s_y \\
\boldsymbol{H}_{3,2} &= 0 \\
\boldsymbol{H}_{3,3} &= m_b + \alpha_m + \alpha_t + 2\alpha_{fl} \\
\boldsymbol{H}_{3,4} &= b_{lx} \alpha_{fl} c_{fl} c_y \\
\boldsymbol{H}_{3,5} &= \frac{b_{ly} \alpha_t \cos(q_t + q_y)}{2} \\
\boldsymbol{H}_{3,6} &= -\alpha_m s_y \\
\boldsymbol{H}_{4,1} &= 0 \\
\boldsymbol{H}_{4,2} &= b_{lx} \alpha_{fl} c_{fl} s_y \\
\boldsymbol{H}_{4,3} &= b_{lx} \alpha_{fl} c_{fl} c_y \\
\boldsymbol{H}_{4,4} &= \frac{2{b_{lx}}^2 \alpha_{fl}}{3}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{H}_{4,5} &= 0 \\
\boldsymbol{H}_{4,6} &= 0 \\
\boldsymbol{H}_{5,1} &= \frac{b_{ly}\alpha_t(4b_{ly} + 3b_{lx}c_t)}{12} \\
\boldsymbol{H}_{5,2} &= \frac{b_{ly}\alpha_t \sin(q_t + q_y)}{2} \\
\boldsymbol{H}_{5,3} &= \frac{b_{ly}\alpha_t \cos(q_t + q_y)}{2} \\
\boldsymbol{H}_{5,4} &= 0 \\
\boldsymbol{H}_{5,5} &= \frac{{b_{ly}}^2\alpha_t}{3} \\
\boldsymbol{H}_{5,6} &= 0 \\
\boldsymbol{H}_{6,1} &= 0 \\
\boldsymbol{H}_{6,2} &= \alpha_m c_y \\
\boldsymbol{H}_{6,3} &= -\alpha_m s_y \\
\boldsymbol{H}_{6,4} &= 0 \\
\boldsymbol{H}_{6,5} &= 0 \\
\boldsymbol{H}_{6,6} &= \alpha_m
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{N}_1 =& 2\alpha_m q_c \dot{q_m}\dot{q_y} + \frac{b_{lx}g\alpha_t c_y}{2} - g\alpha_m q_c c_y + \frac{b_{ly}g\alpha_t c_t c_y}{2} - \frac{b_{lx}b_{ly}\alpha_t {\dot{q_t}}^2 s_t}{4} \\
& - b_{lx}g\alpha_{fl}s_{fl}s_y - \frac{b_{ly}g\alpha_t s_t s_y}{2} + \frac{2{b_{lx}}^2\alpha_{fl}\dot{q_{fl}}\dot{q_y}\sin(2q_{fl})}{3} - \frac{b_{lx}b_{ly}\alpha_t \dot{q_t}\dot{q_y}s_t}{2} \\
\boldsymbol{N}_2 =& \frac{b_{ly}\alpha_t {\dot{q_t}}^2\cos(q_t + q_y)}{2} - 2\alpha_m \dot{q_m}\dot{q_y}s_y + \frac{b_{ly}\alpha_t {\dot{q_y}}^2\cos(q_t + q_y)}{2} + \frac{b_{lx}\alpha_t {\dot{q_y}}^2 c_y}{2} \\
& - \alpha_m q_c {\dot{q_y}}^2 c_y + b_{ly}\alpha_t \dot{q_t}\dot{q_y}\cos(q_t + q_y) - b_{lx}\alpha_{fl}{\dot{q_{fl}}}^2 s_{fl}s_y - b_{lx}\alpha_{fl}{\dot{q_y}}^2 s_{fl}s_y \\
& + 2b_{lx}\alpha_{fl}\dot{q_{fl}}\dot{q_y}c_{fl}c_y \\
\boldsymbol{N}_3 =& gm_b + g\alpha_m + g\alpha_t + 2g\alpha_{fl} - 2\alpha_m \dot{q_m}\dot{q_y}c_y - \frac{b_{ly}\alpha_t {\dot{q_t}}^2\sin(q_t + q_y)}{2} \\
& - \frac{b_{ly}\alpha_t {\dot{q_y}}^2\sin(q_t + q_y)}{2} - \frac{b_{lx}\alpha_t {\dot{q_y}}^2 s_y}{2} + \alpha_m q_c {\dot{q_y}}^2 s_y - b_{lx}\alpha_{fl}{\dot{q_{fl}}}^2 c_y s_{fl} \\
& - b_{lx}\alpha_{fl}{\dot{q_y}}^2 c_y s_{fl} - b_{ly}\alpha_t \dot{q_t}\dot{q_y}\sin(q_t + q_y) - 2b_{lx}\alpha_{fl}\dot{q_{fl}}\dot{q_y}c_{fl}s_y \\
\boldsymbol{N}_4 =& \frac{b_{lx}\alpha_{fl}c_{fl}(3gc_y - 2b_{lx}{\dot{q_y}}^2 s_{fl})}{3}
\end{aligned}
$$

$$\boldsymbol{N}_5 = \frac{b_{ly}\alpha_t(b_{lx}s_t\dot{q_y}^2 + 2g\cos(q_t + q_y))}{4}$$
$$\boldsymbol{N}_6 = -\alpha_m(q_c\dot{q_y}^2 + gs_y)$$

Table A.1: Parameters for bat robot

State	Units	Value
b_{lx}	m	0.1
b_{ly}	m	0.04
b_{lz}	m	0.04
c_1	m	0.01
m_b	g	50

REFERENCES

[1] N. Kato and S. Kamimura, *Bio-Mechanisms of Swimming and Flying: Fluid Dynamics, Biomimetic Robots, and Sports Science*. Springer, 2008.

[2] A. Ramezani, X. Shi, S.-J. Chung, and S. Hutchinson, "Bat bot (b2), a biologically inspired flying machine," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[3] J. Hoff, A. Ramezani, S.-J. Chung, and S. Hutchinson, "Synergistic design of a bio-inspired micro aerial vehicle with articulated wings," in *The Robotics: Science and Systems Conference (RSS)*, Ann Arbor, Michigan, June 2016.

[4] U. Syed, A. Ramezani, S.-J. Chung, and S. Hutchinson, "From rousettus aegyptiacus (bat) landing to robotic landing: Regulation of cg-cp distance using a nonlinear closed-loop feedback," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[5] D. K. Riskin, D. J. Willis, J. Iriarte-Díaz, T. L. Hedrick, M. Kostandov, J. Chen, D. H. Laidlaw, K. S. Breuer, and S. M. Swartz, "Quantifying the complexity of bat wing kinematics," *Journal of Theoretical Biology*, vol. 254, no. 3, pp. 604–615, Oct. 2008.

[6] A. J. Bergou, S. M. Swartz, H. Vejdani, D. K. Riskin, L. Reimnitz, G. Taubin, and K. S. Breuer, "Falling with style: Bats perform complex aerial rotations by adjusting wing inertia," *PLOS Biology*, vol. 13, no. 11, pp. 1–16, Nov. 2015.

[7] C. Y. Brown and H. H. Asada, "Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 2877–2882.

[8] A. Paranjape, S.-J. Chung, and J. Kim, "Novel dihedral-based control of flapping-wing aircraft with application to perching," *IEEE Transactions on Robotics*, vol. 29, pp. 1071–1084, October 2013.

[9] P. Holmes, R. J. Full, D. Koditschek, , and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges," *SIAM Review*, vol. 48, no. 2.

[10] I. Poulakakis and J. W. Grizzle, "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1779–1793, Aug 2009.

[11] K. Sreenath, H.-W. Park, I. Poulakakis, , and J. W. Grizzle, "Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on mabel," *The International Journal of Robotics Research*, vol. 22, pp. 988–1005, June 2014.

[12] N. A. Bernstein, *The coordination and regulation of movements*. Oxford: Pergamon Press, 1967.

[13] M. Santello, M. Flanders, and J. F. Soechting, "Postural hand synergies for tool use," *Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, 1998. [Online]. Available: http://www.jneurosci.org/content/18/23/10105

[14] C. Y. Brown and H. H. Asada, "Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 2877–2882.

[15] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press: John Wiley and Sons, Inc., 2012.

[16] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Philadelphia, PA: SIAM Advances in Design and Control, 2010.

[17] R. Tedrake and the Drake Development Team, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2016. [Online]. Available: http://drake.mit.edu

[18] P. E. Gill, W. Murray, M. A. Saunders, and E. Wong, "User's guide for SNOPT 7.6: Software for large-scale nonlinear programming," Department of Mathematics, University of California, San Diego, La Jolla, CA, Center for Computational Mathematics Report CCoM 17-1, 2017.

[19] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, pp. 338–342, July 1987.

[20] E. Hairer and G. Wanner, Eds., *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, ser. Computational Mathematics. Springer, 1996, vol. 14.

[21] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. New York: John Wiley and Sons, Inc, 2005.

[22] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832)," 2016. [Online]. Available: http://underactuated.csail.mit.edu/

[23] L. F. Shampine and M. W. Reichelt, "The matlab ode suite," *SIAM Journal on Scientific Computing*, vol. 18, pp. 1–22, 1997.