Simulations of a Three Degree of Freedom Brachiating Y-Bot Robot

by

Alexander Breton

Submitted to the
Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2016

Signature of Author: Signature redacted

Alexander G. Breton
Department of Mechanical Engineering
May 17, 2016

Certified by: Signature redacted

Sangbae Kim
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: Signature redacted

Anette Hosoi
Professor of Mechanical Engineering
Undergraduate Officer

Simulations of a Three Degree of Freedom Brachiating Y-Bot Robot

by

Alexander Breton

Submitted to the Department of Mechanical Engineering
on May 17, 2016 in Partial Fulfillment of the
Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

ABSTRACT

Brachiation is a means of locomotion for lightweight apes like gibbons. It involves the animal swinging its arms to gain moment and swing forward. A large amount of research has been done studying a simplified two-link two DOF robot, named "acrobot" by Mark Spong. While the problems of this robot have been studied extensively, it's functionality is quite limited. This paper studies a three-link three DOF brachiating robot, dubbed "Y-bot". The goal of adding the extra link is to add functionality.

Simulations of a model were run in Matlab taking advantage of Russ Tedrake's toolbox Drake, which was designed to solve optimization problems of underactuated systems. The main method used in the trajectory optimization was direct collocation. The task of the robot in the simulations was to swing from a one "branch" point to another. The trajectories of two Y-bot models swinging from rest were optimized. Furthermore, the gait of one of the models was examined, and a beneficial state for the second swing of a gait was suggested. A method to optimizing the gait of a model was proposed. A linear relationship between the total trajectory time and the scale of the model was defined.

The paper suggests a physical model of the Y-bot could be constructed using Saito's two DOF brachiating robot as a benchmark. The problems of gait optimization and payload transportation were mentioned as future work to be done.

Thesis Supervisor: Sangbae Kim
Title: Associate Professor of Mechanical Engineering

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Brachiation Description

Brachiation is a means of locomotion for animals which involves an arm under arm swinging motion. Arguably the best animals at brachiating are gibbons, smaller long-limbed apes from Southeast Asia. Compared to other apes, these gibbons have relatively light bodies, which makes brachiation a suitable means of locomotion for them. Through brachiating, they are able to use the momentum of their falling arm to swing them forward from branch to branch. All apes are able to brachiate to some level, but because branches often will not support the heavy bodies of the larger apes, the larger animals are rarely seen moving in this way. Even humans, although we have developed smaller muscles in our upper bodies, are able to brachiate (like a child playing on monkey bars) [1]. This slower swinging can be seen in Fig 1.1. In certain cases, gibbons have been seen flinging themselves forward through the air to move through the forest faster, while maintaining a similar rhythm and arm under arm motion as the slower movement [2].

1.2 Previous work

Brachiating robots that have been developed by multiple researchers are typically simplified two degree of freedom robots consisting of two links connected by a revolute joint. A diagram of this model can be seen in Fig. 1.2. The main actuator in the robot is fixed to this 'elbow' joint, which actuates the second joint. Grippers are attached to the ends of each link to grip on bars or simulated branches. In experiments, one of the arms grips onto a bar, freely swinging on it, while the other arm is actuated at the elbow joint. For this brachiation movement, the robot must take advantage of both the kinematic energy and the potential energy properly in order to make the swing to the next desired position. It is important to note that since there is only one actuator controlling the movement of the robot and the robot has two degrees of freedom, the system is considered underactuated. Underactuated systems provide a host of interesting challenges that do not come about in fully actuated systems. The first group of researchers to delve into this brachiation problem were Saito *et al.* In 1994, the group built a physical model of the two DOF system mentioned above [3]. A schematic of their robot is shown in Fig 1.3. In one of their papers, they developed a feed forward controller for swinging from

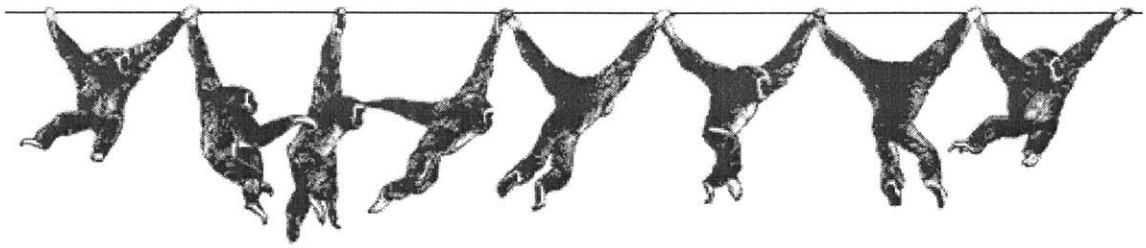

Figure 1.1: A diagram of a gibbon brachiating, from [4]

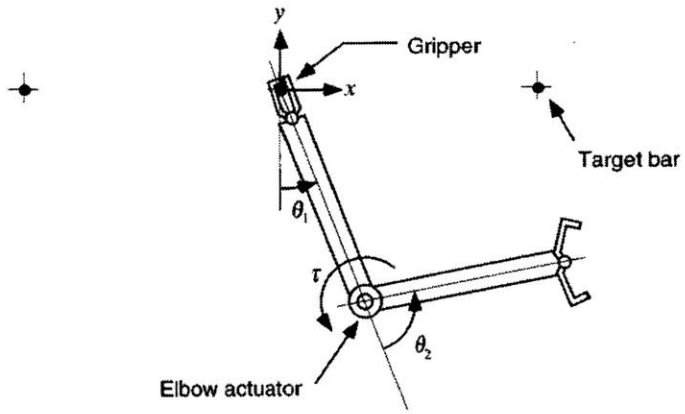

Figure 1.2: A diagram of the Nakanishi *et al.* 2 DOF brachiating robot, from [6]

one 'branch' to the next. While this method proved effective, it required an extensive trial and error process, requiring over 200 experiments [5]. Researchers in the field like Nakanishi, Fukuda, and Koditschek thus began to develop feedback controllers to solve various problems for these brachiation robots [6]. Spong *et al.* also began working on a controller for the swing-up problem for the simplified brachiation robot, which they named the "acrobot" [7]. The swing-up problem requires the robot to swing up to a branch from a hanging position in which only one gripper is holding a bar. Later, Nakanishi defended his PhD thesis on a developed controller [8].

In 2006, Menezes and Lages [9] began looking at a 3 DOF that would be able to inspect power lines for failures. They developed a controller implementing nonlinear model-based

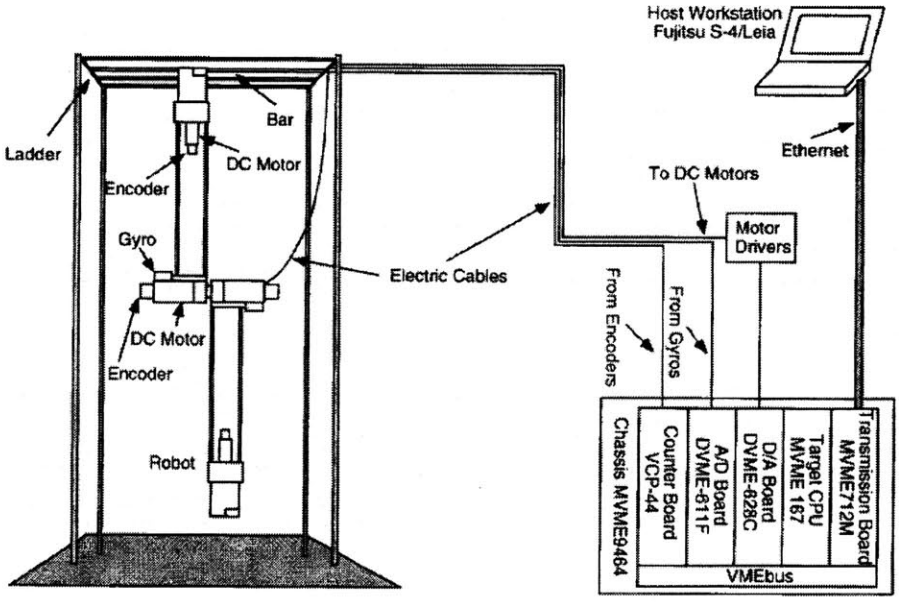

Figure 1.3: The schematic of Saito's 2 DOF Brachiating Robot, from [3]

predictive control (NMPC) with some success. They found, however, that it was not possible to use the nonlinear model in real time because of the computational demands. Therefore, they continued their work and switched to a linear MP, which worked effectively in real time [10]. Fukada et al. [11] examined a much more complex multi-locomotion 24 DOF monkey robot which was able to walk bipedally, walk quadrapedally, and brachiate. Still, for brachiation purposes, the system was able to be simplified to a 3 DOF model similar to Menezes and Lages' model. For control purposes, they used Passive Dynamic Autonomous Control (PDAC). They were able to have the robot continuously brachiate, but they also used a more complex two phase locomotion consisting of a stationary swinging action followed by a swinging locomotion action.

In 2012, Askeland [12] did similar work, creating a controller for the simplified model of the 24 link robot. He designed a MATLAB toolbox based on the theory of orbital stabilization. The controller was proven to be effective for an inverted pendulum system, but Askeland explained that further work needed to be done to apply it to the brachiating robot. Recently, a group out of the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), led by Russ Tedrake, has developed a different MATLAB toolbox, Drake, that is dedicated to solving problems like those created by underactuated systems. This powerful toolbox was implemented in MIT's humanoid robot entered into the DARPA Robotics Challenge [13].

1.3. 3 DOF Brachiation Robot

In this paper, I present results from simulations of a 3 degree of freedom brachiating robot similar to the acrobot mentioned above. This 3 DOF robot, or "Y-bot", has the two "arms" of the acrobot, with an added link from the elbow acting as the "body." This Y-bot is similar to the models used in some of the work mentioned above [6,10,11,12]. This extra link is also unactuated, meaning there are now two unactuated links and only one actuated link. While 2 DOF brachiation robots provide a good testbed for controls, their functional abilities are limited. The added link would provide an opportunity for added functionality of the robot. The body link could be used to simply carry a payload that needed to be transported with the robot. A more creative concept could involve the free swinging body link actually being another acrobot, with a fixed shoulder at the Y-bot elbow replacing one of the grippers. Once the Y-bot grippers are holding the robot in the desired position, the body link acrobot would be able to swing up to a separate desired point. Menezes and Lages suggested that a similar 3 DOF robot could be used to inspect power transmission lines [6,10]. Most importantly, this paper suggests a method of simulation for continuous brachiation of the simplest Y-bot model, which has not been examined in the works mentioned above. The simulations were completed using the Drake toolbox.

2. Set Up

2.1 Model

The model in Fig. 2.1. was used for the dynamics of this system and the equations of motion were derived using the Lagrangian method. The parameters of the degrees of freedom are as follows: θ_1 corresponds to the shoulder joint angle, θ_2 corresponds to the angle of the actuated link 2 relative to link 1, and θ_3 corresponds to the angle of the unactuated body link 3 relative to link 1. These angles will be expressed as the matrix $\boldsymbol{q} = [\theta_1\ \theta_2\ \theta_3]^T$, and the state of the system will be expressed as the matrix $\boldsymbol{x} = [\boldsymbol{q}\ \dot{\boldsymbol{q}}]^T$. The kinetic energy, $\boldsymbol{T}$, and potential energy, $\boldsymbol{V}$, were found and plugged into the Lagrangian equations:

$$\boldsymbol{L} = \boldsymbol{T} - \boldsymbol{V}$$

$$\frac{d}{dt}\frac{\delta \boldsymbol{L}}{\delta \dot{\boldsymbol{q}}} - \frac{\delta \boldsymbol{L}}{\delta \boldsymbol{q}} = \boldsymbol{Q}$$

Where the generalized force vector is $\boldsymbol{Q} = [0\ \tau\ 0]^T$ due to the non-actuated links. The equations of motion can be rearranged into the manipulator equations format:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{u}$$

Where $\boldsymbol{H}$ is the 3x3 inertia matrix, $\boldsymbol{C}$ is the 3x3 Coriolis matrix, $\boldsymbol{g}$ is the 3x1 gravity matrix, and $\boldsymbol{u}$ is the 3x1 control input vector $[0\ \tau\ 0]^T$. Here, for ease of transcription, trigonometric functions will be written as $sin(\theta_1) = s_1$ and $cos(\theta_1) = c_1$. The elements of $\boldsymbol{H}$ are calculated as follows:

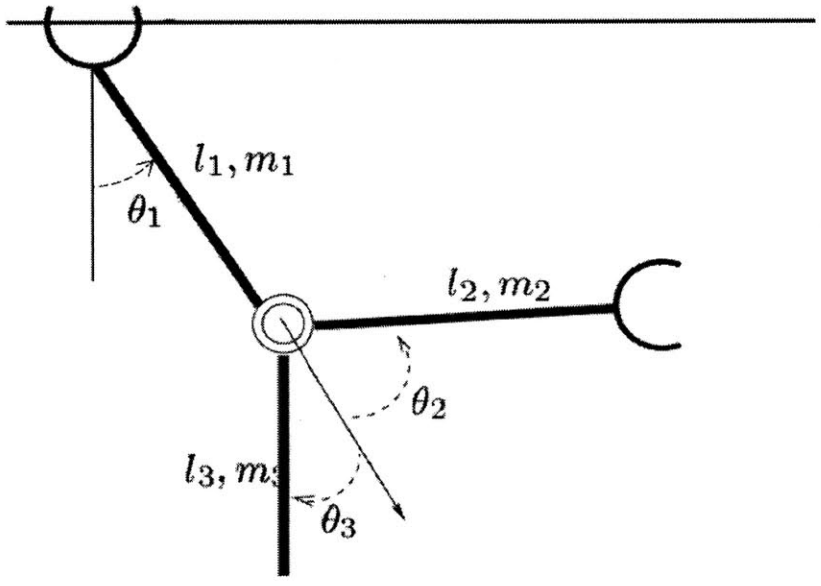

Figure 2.1: The diagram of the Y-bot, modified from []

$$\boldsymbol{H}_{11} = I_1 + I_2 + I_3 + m_2 {l_1}^2 + 2m_2 l_1 l_{c2} + m_3 {l_1}^2 + 2m_3 l_1 l_{c3} c_3$$

$$\boldsymbol{H}_{12} = \boldsymbol{H}_{21} = I_2 + m_2 l_1 l_{c2} c_2 \qquad \boldsymbol{H}_{13} = \boldsymbol{H}_{31} = I_3 + m_3 l_1 l_{c3} c_3$$

$$\boldsymbol{H}_{22} = I_2 \qquad \boldsymbol{H}_{23} = \boldsymbol{H}_{32} = 0 \qquad \boldsymbol{H}_{33} = I_3$$

The elements of $\boldsymbol{C}$ are calculated as follows:

$$\boldsymbol{C}_{11} = -2m_2 l_1 l_{c2} s_2 \dot{q}_2 - 2m_3 l_1 l_{c3} s_3 \dot{q}_3$$

$$\boldsymbol{C}_{12} = -m_2 l_1 l_{c2} s_2 \dot{q}_2 \qquad \boldsymbol{C}_{13} = -m_3 l_1 l_{c3} s_3 \dot{q}_3$$

$$\boldsymbol{C}_{21} = -m_2 l_1 l_{c2} s_2 \dot{q}_1 \qquad \boldsymbol{C}_{31} = -m_3 l_1 l_{c3} s_3 \dot{q}_1$$

$$\boldsymbol{C}_{22} = \boldsymbol{C}_{23} = \boldsymbol{C}_{32} = \boldsymbol{C}_{33} = 0$$

The elements of $\boldsymbol{g}$ are calculated as follows:

$$\boldsymbol{g}_{11} = g[m_1 l_{c1} s_1 + m_2 (l_1 s_1 + l_{c2} s_{12}) + m_3 (l_1 s_1 + l_{c3} s_{13})]$$

$$\boldsymbol{g}_{21} = m_2 l_{c2} s_{12} \qquad \boldsymbol{g}_{31} = m_3 l_{c3} s_{13}$$

It is important to note that the moments of inertia here are taken about the joint. The moment of inertia I_c is the moment of inertia about the center of mass. The moment of inertia around the joint of a link i can be calculated using the parallel axis theorem:

$$I_i = I_{ci} + m_i {l_{ci}}^2$$

2.2. Nonlinear Optimal Trajectory Planning Problem

Nonlinear programming is a solving an optimization problem which has either nonlinear constraints or a nonlinear objective function which is trying to be minimized. These constraints are a set of equalities or inequalities dealing with the variables in the objective function. Usually this objective function is some sort of cost function. A standard cost function is the additive cost model:

$$J = \int_0^T g(\boldsymbol{x}(\boldsymbol{t}), (\boldsymbol{u}(\boldsymbol{t}))dt$$

Where *g()* is the cost at a given time. The goal is to find the control trajectory that minimizes the cost over the time specified by the limits of integration. If the problem is concave or convex (i.e. the local minima or maxima are global minima or maxima), then simpler convex optimization techniques may be used. If, however, the problem is not convex, different strategies must be employed [14].

2.2.1 Direct Linear Collocation

The method of linear collocation was first suggested by Hargaves in 1987 [15]. He stated that "the primary advantage of this method is that it is much easier to extend to general problems

involving [...] state and control inequalities." The direct collocation method is a way of solving the ordinary differential equations of the problem numerically, and is a type of Runge-Kutta method. To start, the system must be described in the following form:

$$\boldsymbol{x}'(t) = f(\boldsymbol{x}, \boldsymbol{u}, t) .$$

A finite time interval is set and broken into smaller intervals at N break points $c_1, c_2, \ldots, c_N$. At the beginning of each interval $[t_{Ci}\ t_{Ci+1}]$, the intial condition is

$$\boldsymbol{x}(t_{C_i}) = \boldsymbol{x}_i .$$

It is also possible to impose equalities and inequalities onto the system. The goal of this method is to find the trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ that minimize a given cost function while satisfying the constraints and equations above.

This direct collocation method approximates the state at each interval as a cubic polynomial and the control at each interval as a linear polynomial which satisfy the conditions at the break points. Both polynomials are described by the values and derivatives at the collocation points. The derivative of the polynomial at the midpoint of each segment is then taken and compared to the system's dynamics at that point. The goal is to minimize the difference between the two values in order to obtain an accurate solution approximation. Once the state and control trajectories are completely implicitly defined, they can be minimized according to the cost function [14,15,16].

2.2.2. Implementation in Matlab through Drake

First, a subclass Y-bot is made under the Drake subclass, *Manipulator*. The model is defined as a 3 DOF *Manipulator* object with the parameters, dynamics, and equations of motion as calculated above. A separate function within the Y-bot class is made to determine the optimal trajectory via direct collocation. The Y-bot object is defined as an object in the class *DircolTrajectoryOptimization*, a subclass of *DirectTrajectoryOptimization*. Each subclass under *DirectTrajectoryOptimization* must have both a running cost function and dynamic constraints. The subclass also has parameters N and *durations* which are the number of time samples and the total trajectory time bounds, respectively.

Once the object has been created, state constraints for the initial and final time steps are added using the function *addStateConstraint*. At $t = 0$, the equality state constraint is implemented as $\boldsymbol{x} = [-\pi/4, -\pi/2, \pi/4, 0, 0, 0]^T$, specifying the type with the function *ConstantConstraint*. This state corresponds to both grippers holding onto bars a distance $\sqrt{2}$ times longer than the links, while the body link hangs freely. All the links are at rest. This position can be seen in Fig 2.2. At $t = t_f$, the inequality state constraint is $[\pi/4, \pi/2, -\infty, 0, 0, -\infty]^T < \boldsymbol{x} < [\pi/4, \pi/2, \infty, 0, 0, \infty]^T$, specifying the type with the function *BoundingBoxConstraint*. This constraint is essentially only an inequality constraint for the body link, while the two arm links are effectively subject to an equality constraint. The desired positions and velocities of the two

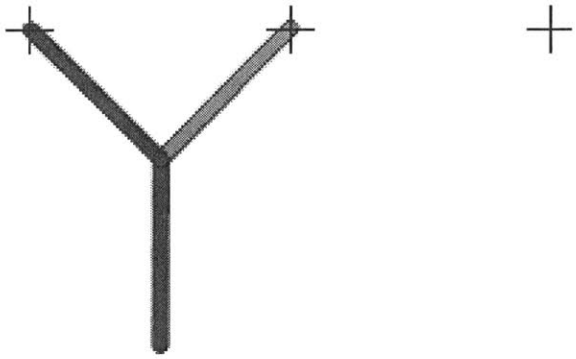

Figure 2.2: Y-bot at initial position $\mathbf{x} = [-\pi/4, -\pi/2, \pi/4, 0, 0, 0]$

arms is known: they should be mirrored from the start positions about the vertical axis. The position and velocity of the body, however, is left unbounded because the link's final state can be arbitrary due to the formulation of the problem. Leaving the state unbounded allows us to see the behavior of the link as the other two links attempt to reach the goal state. For the running cost of the system, the following function was defined, with the goal to minimize control:

$$J = \int_{t_i}^{t_f} \boldsymbol{u}^T \boldsymbol{R}\, \boldsymbol{u}\, dt$$

where $\boldsymbol{R}$ is a gain matrix.

Now with the object adequately defined, the function *SolveTraj* in *DirectTrajectoryOptimization* uses a solver to solve the nonlinear problem and output the optimized state and control trajectories. The solver used in these simulations is *SNOPT*, a piece of software developed at Stanford [17] . Once the problem has been solved, a subclass of the *Visualizer* class is run to create a simple visual of the model moving through the specified trajectory. The state variables and control values are also plotted against time.

3. Simulation and Results

The goal of these simulations was to examine the brachiating behavior of a 3 DOF Y-bot, a robot very similar to the 2 DOF acrobot. To gain some intuition on the two robots, data from passive simulations (i.e. with no control) can be examined. When the acrobot is released from a starting position as seen in Fig 2.2., the two links begin to swing and almost instantly become in phase with one another. The effect of the Y-bot's extra link can be seen in its passive simulation. When the Y-bot is released from a starting position mentioned above in Fig. 3.1.a., the arm that released its grip begins to swing like the Acrobot but is jolted out of its smooth trajectory by the inertia of the body link starting. That arm and the body link then begin to swing almost perfectly out of phase, while the other arm begins to swing with a period 2.5 times the periods of the other links' swings (see Fig. 3.1.b.).

3.1. Characteristics of the Equilink Model

The first set of simulations used a Y-bot consisting of 3 links with the same parameters, hereafter referred to as the equilink model (see Table 3.1. for the parameters of this simulations model). The goal of the simulation was to find the optimal trajectory that minimized both time and control. In previous research with acrobots, the desired path was a mirrored trajectory that had the arms evenly swing after release to the next point [citation]. A similar trajectory which consisted of a single fluid swing was hypothesized to also be the optimal trajectory for the Y-bot. After some preliminary simulations, the time constraints on the trajectory solver were set to limit the trajectory to one fluid swing. The result was a trajectory that reaches its goal state at $t_f = 1.65$ s. The final state of the third link is $\boldsymbol{q_3} = [-0.5506\ 2.1031]$. The trajectory of the three links and the control torque plotted against time can be seen in Fig. 3.2.

3.1.1. Continuous Brachiation

One aspect of the Y-bot to examine is continuous brachiation. An acrobot can easily swing with a continuous and even gait because each end position of the swing is identical to the beginning of the swing. The Y-bot, however, has the uncontrolled body link. The final state of

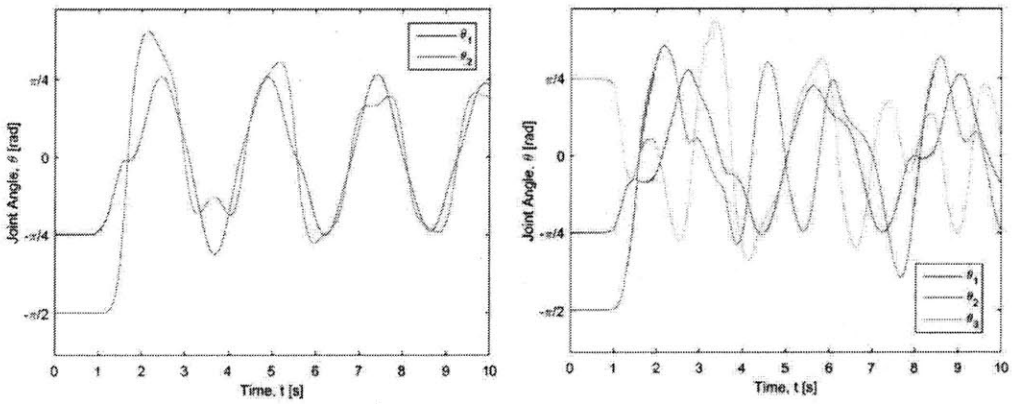

Figure 3.1: a) The passive trajectory of an acrobot. b) The passive trajectory of the Y-bot

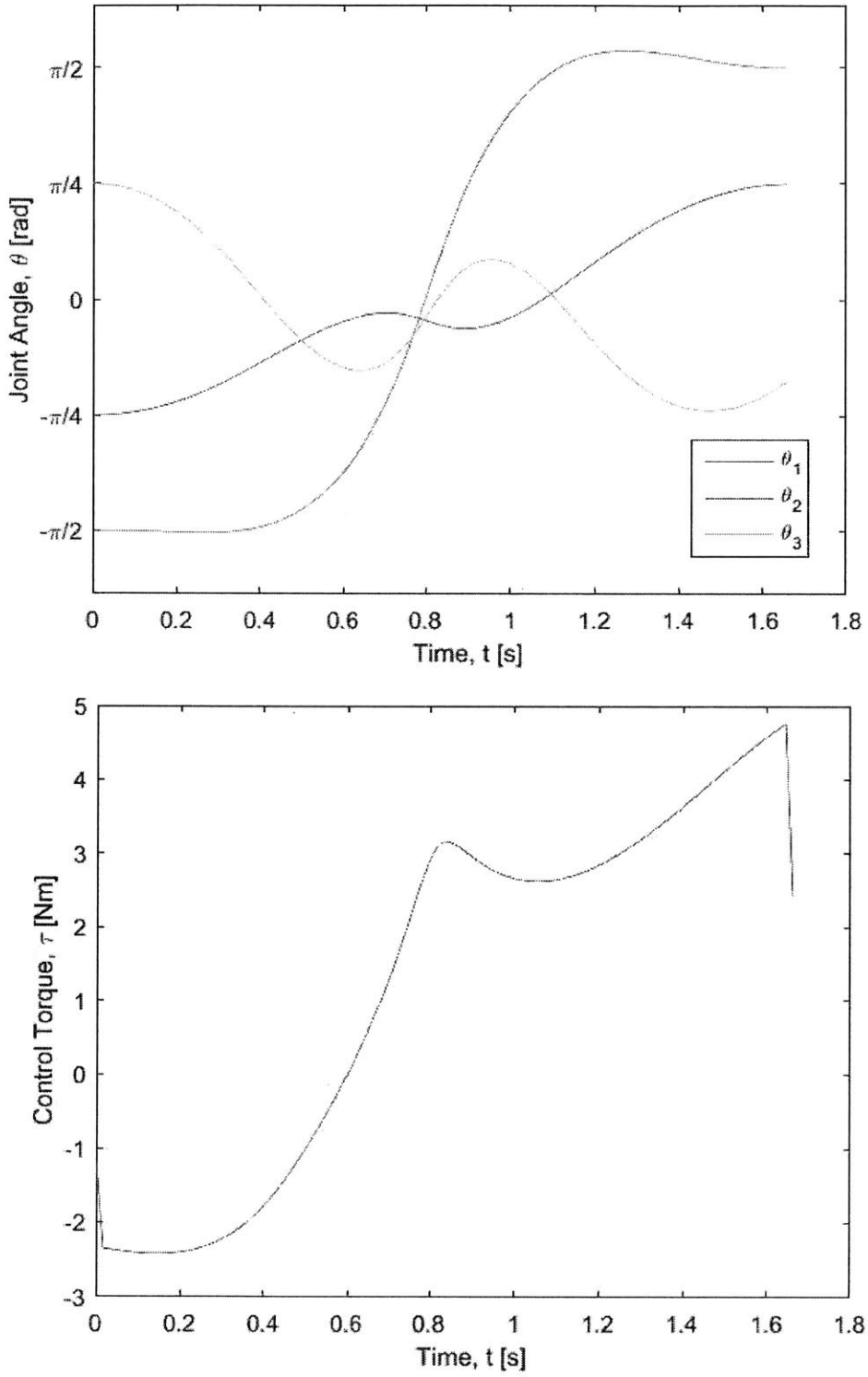

Figure 3.2: Trajectory and torque of the optimal first swing of the equilink model plotted against time

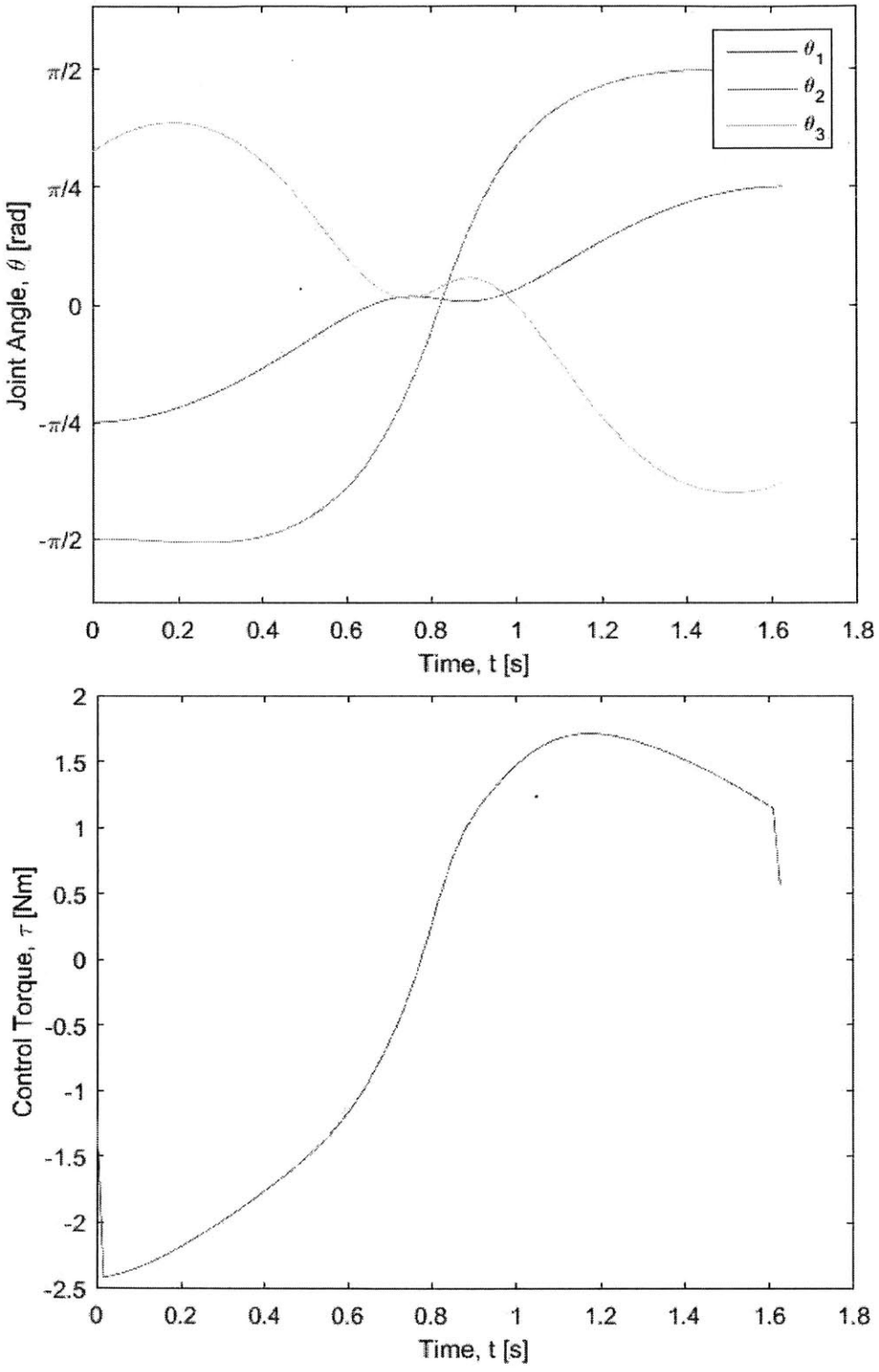

Figure 3.3: Trajectory and torque of the second swing of the equilink model, starting from the optimal state

Mass, m	1 kg
Length, l	2 m
Moment of Intertia at COM, I_C	0.33 kg·m^2
Distance to COM, l_c	1 m
Viscous friction, b	.1 N·m/s

Table 3.1: The values of the parameters of the each link of the Y-bot used in the equilink simulation.

this extra link must be taken into consideration as it determines what the initial state of the next swing will look like. The problem now is to find the optimal starting state of the link for the Y-bot to begin the next swing. For example, if the simulation above began the second swing *0.3* s after grasping, the robot would not physically be able to swing to the next point in one fluid motion. Rather, it would swing forward a certain distance, swing back, and then swing fully up. This trajectory, however, is not type of optimal trajectory as discussed above. A simple inequality constraint cannot just be applied to the initial state. Rather, the optimal state must also be chosen from a state that will occur in the model. Once the robot reaches the final state of a swing, the body link can be simply modelled as a physical pendulum. The trajectory from which the initial states may be chosen from can be found through a pendulum simulation. One period of the trajectory of the body link after the first swing of the equilink simulation is shown in Fig. 3.4. The initial state of the next swing of the simulation must thus be chosen from this trajectory. The problem now becomes determining the optimal time after gripping to begin the next swing. To determine this value, points along the trajectory were chosen and set as the initial state of new swing simulations, and the total swing times and impulses were collected. The plots of these data can be seen in Figs. 3.5 and 3.6., respectively. After the first swing, the optimal time to begin the second swing is at immediately after the first swing is completed. This choice optimizes both time and control. The trajectory and control torque for this optimal second swing can be seen in Fig. 3.3.

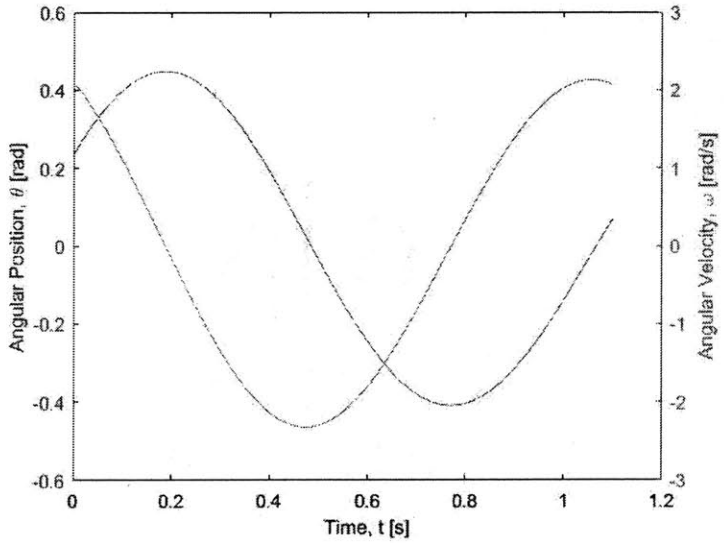

Figure 3.4: Position and velocity trajectory of the body link after the first swing of the equilink model simulation, acting as a pendulum

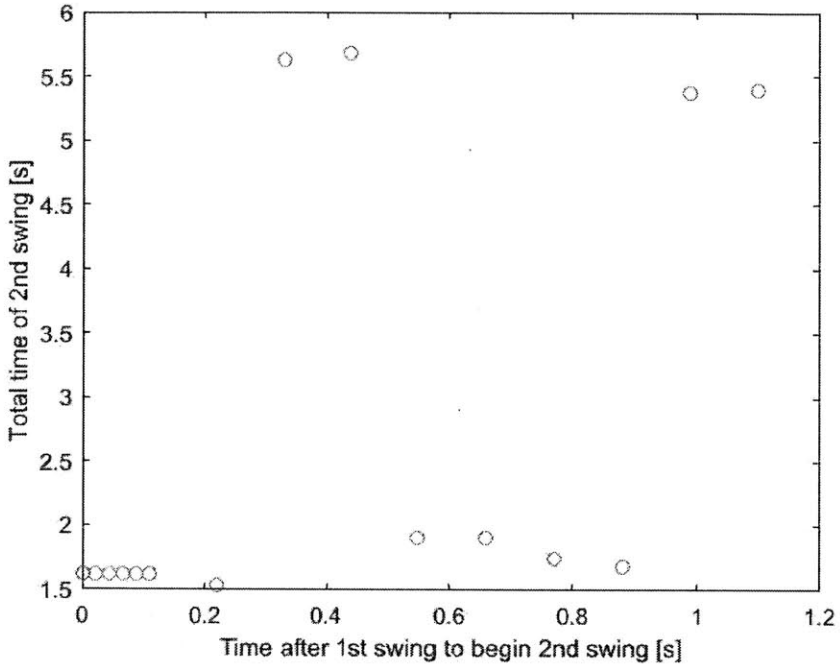

Figure 3.5: The total time the 2nd swing would take is plotted against the time after the 1st swing is completed that the 2nd swing would begin. There are two clear groupings of data: those closer to 2 seconds represent simulations that swing up in one fluid motion, while the longer simulations involve kipping up.

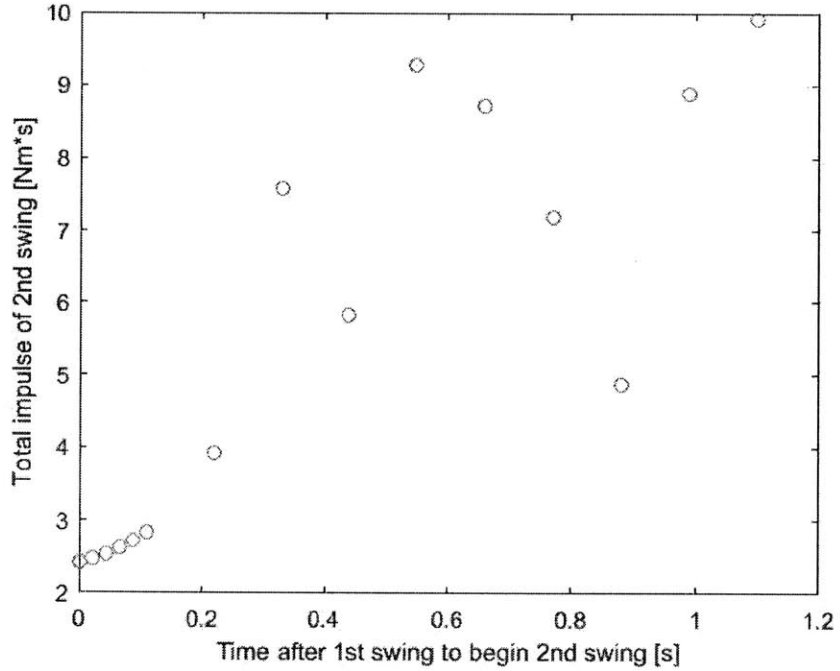

Figure 3.6: Total impulse of the 2nd swing for each simulation is shown. The optimal choice in terms of minimizing control is at $t = 0$, or immediately after the 1st swing.

3.2. Body Link as Smaller Acrobot

As mentioned above, a potential use of the body link could be a smaller Acrobot that is able to complete specialized tasks. This link can be modeled as the previous link with a point mass at the center to represent the elbow actuator. For this model, the body mass is doubled, while the moment of inertia remains the same as the equilink model. The same simulation was run with the new model, and the results can be seen in Fig 3.7. The trajectory is very similar to the equilink optimal trajectory, but here the body link swings less due to its higher moment of inertia.

3.3 Optimal Trajectory Time as a function of length

When the parameters of the links in the equilink model are evenly scaled, the shape of the optimal swing trajectories are very similar. The time to complete the swing up also increases as seemingly linearly. A comparison of the trajectories of scaled equilink models can be seen in Fig. 3.8., and the final swing times plotted against the parameter scaling in Fig. 3.9.

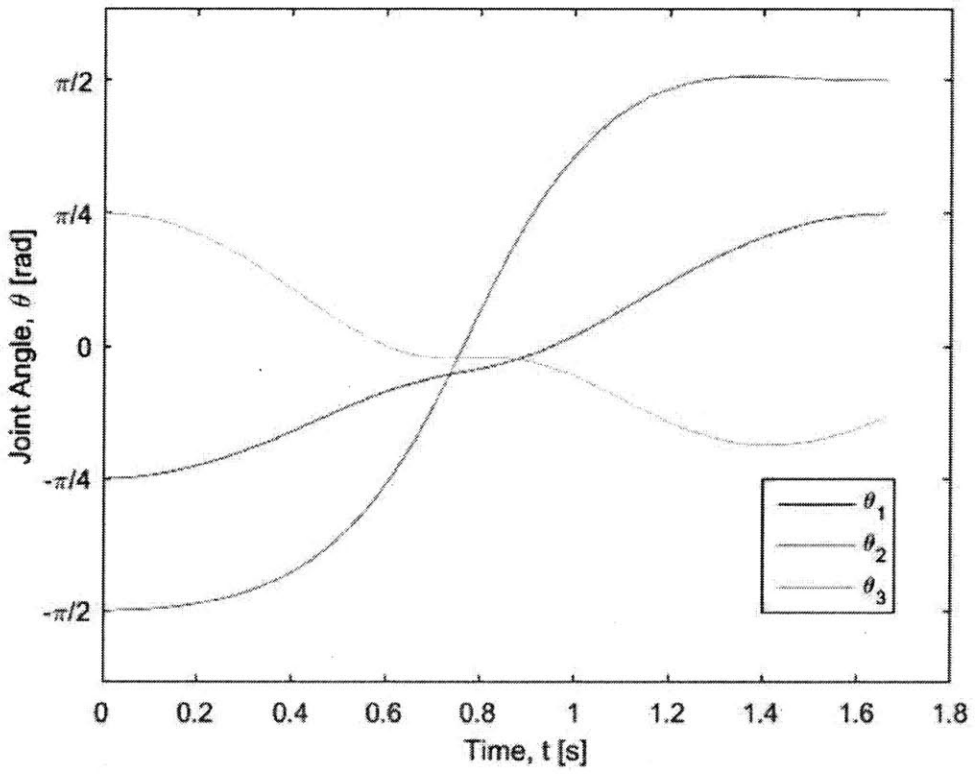

Figure 3.7: Optimal Trajectory of a Y-bot with an acrobot body link

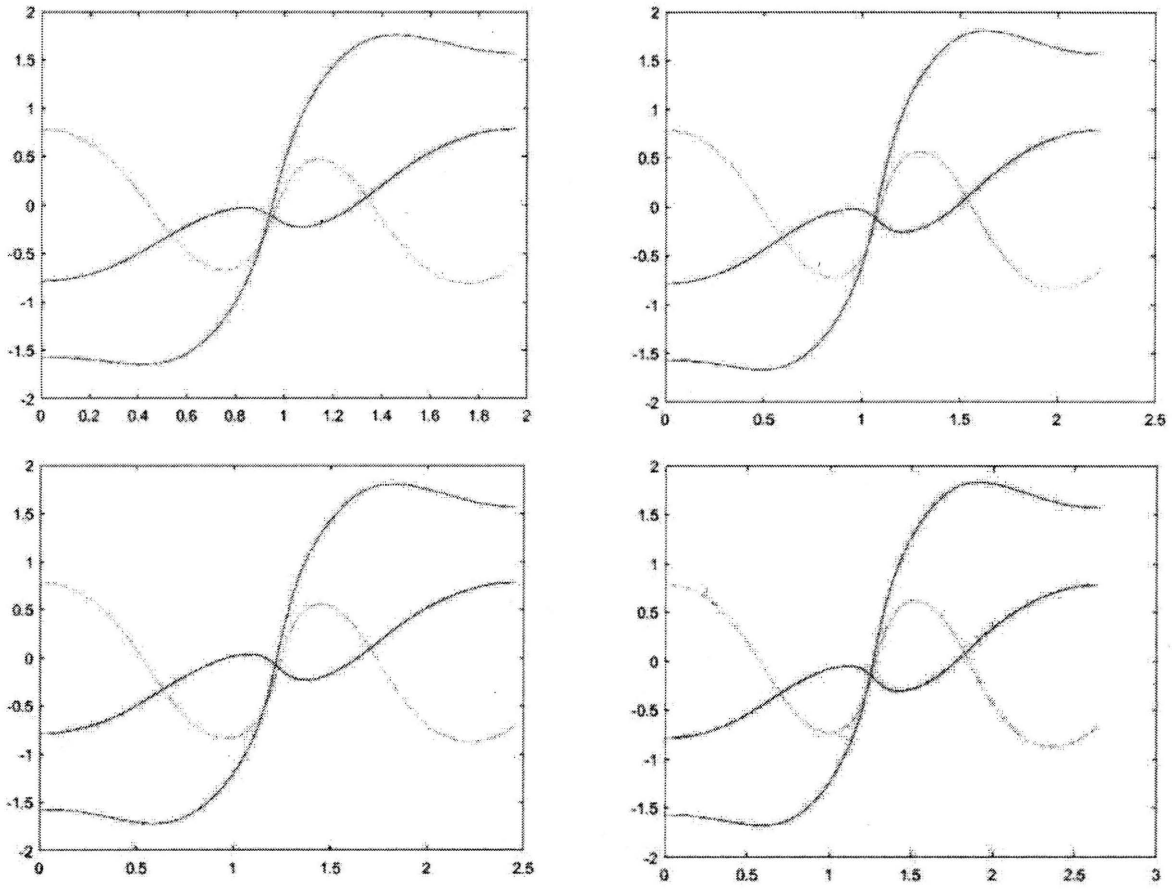

Fig 3.8: Optimal trajectory swings of equilink with parameters scaled by a factor of *k*. a) *k=1.5* b) *k=2* c) *k=2.5* d) *k=3.*

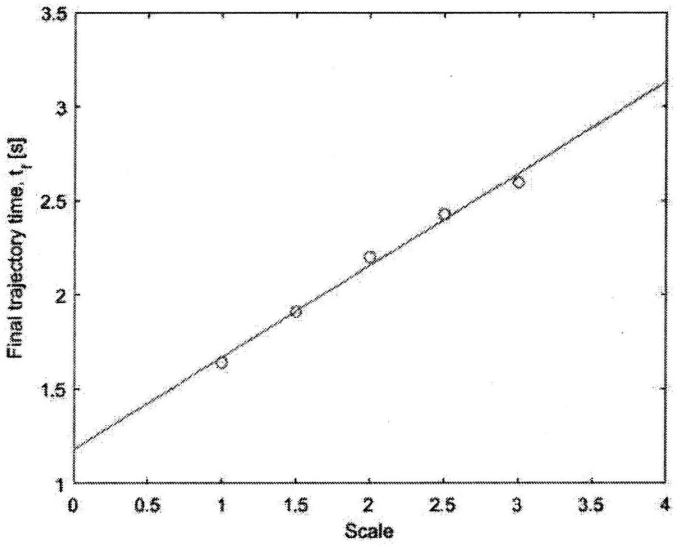

Fig 3.9: Final trajectory times of scaled equilink simulations plotted against scale constant. Fit with a linear polynomial

4. Conclusion

These experiments examined the behavior of the Y-bot during brachiation from one "branch" to another. An optimal trajectory for the equilink model was found for a swing with the robot starting at rest. The following swings of a gait were considered, and a beneficial starting state was found for the second swing. An outline of the method of choosing a good starting state was also discussed. An optimal trajectory for the embedded Acrobot model was found as well, proving to be very similar to the equilink's optimal trajectory. Finally, the swing up times of differently scaled equilink models were described as a linear function of scale.

4.1. Open Problems

The biggest problem that remains is finding a method that enables the robot to choose at which state to begin its swing mid-gait. This paper has provided a very rough method but was solved manually, piece by piece, and only answers the question for the second swing. This problem becomes much more important as the Y-bot is required to traverse longer distances. An ideal solution would be able to gather the end state from the previous swing, calculate the body link's trajectory, and choose the optimal point in the trajectory to begin the next swing. This solution would theoretically be able to be applied across all Y-bot models

Another open problem is finding an optimal trajectory for a payload-carrying Y-bot. Some simulations were run modelling the body link as a point mass pendulum with the mass and length of an arm link. Because the moment of inertia was too high, the robot was not physically able to reach the goal state in one swing up. This could be solved by either making the pendulum shorter or lightening the load. If the pendulum is shortened, the payload's volume and shape would be restricted by the geometry of the robot. If the load was lightened, it would point to a decrease in functionality of the system. The question of the Y-bot's efficacy of carrying loads remains to be answered.

4.2. Future steps

Nakanishi *et al.* [6] used a successful physical model design for the Acrobot, shown in Fig. 1.3. This model implements two servo motors coupled together at the elbow to provide twice the toque and maximum speed of a single motor, and also to balance the weight of the robot. This model could potentially be turned into a Y-Bot by adding another link. If the coupling between the motors was lengthened to increase the distance between the motor, the body link could be mounted on that coupling, essentially functioning as an axle. The body link would be between the two arms, which accurately reflects the biological model of the gibbon. The end of the body link mounted on the coupling would be a fixed shape shoulder with bearings rather than a gripper. The other end could accommodate a gripper, a payload, or another functional attachment. As the length of the coupling is increased, Abbe error effects on the motors must be taken into effect and accounted for. The coupled shafts will act as a cantilevered beam with the weight of the hanging arm acting as a downward force on the end. The deflection

downward will need to be minimized to maintain parallelism between the links. Bearing will also need be carefully selected and placed to protect the motors from transverse loads. Once the physical model is built, a controller must also be designed. One solution could be to implement a time-varying linear quadratic regulator (LQR). Great care must be taken in the representation of the physical system in the model for the LQR to be useful because there are 3 DOF with only one actuated DOF. Perhaps an even better controller than a general LQR would be one of the controllers from the research mentioned above which are specifically tailored to brachiation robots.

6. Bibliography

[1] O'Neill, Dennis. "The Primates: Apes." *The Primates: Apes*. Palomar College, n.d. Web. 3 Apr. 2016.

[2] *Gibbon Swinging through the Trees Very Fast. Youtube*. N.p., 13 Feb. 2011. Web.

[3] Saito, Fuminori, Toshio Fukuda, and Fumihito Arai. "Swing and Locomotion Control for a Two-Link Brachiation Robot." *IEEE Control Systems Magazine*, 12:5-12, February 1994.

[4] Gibbons Swinging. Digital image. *Locomotion*. Gibbon Research Lab,. Web.

[5] Saito, Fominori. , "Motion control of the brachiation type of mobile robot," Ph.D. dissertation (in Japanese), Nagoya Univ., Nagoya, Japan, Mar. 1995.

[6] Nakanishi, J., Fukuda, T., and Koditschek, D.E. "Preliminary studies of a second generation brachiation robot controller". April 1997.

[7] Spong, M. "The swing up control problem for the acrobot." *IEEE Control Systems Magazine,* 15:49-55, February 1995.

[8] Nakanishi, Jun. Thesis. Nagoya University, 2000.

[9] V. M. de Oliveira and W. F. Lages, "Predictive Control of an Underactuated Brachiation Robot," *IFAC Proceedings Volumes, 8th IFAC Symposium on Robot Control*, 2006, pp. 19-24.

[10] V. M. de Oliveira and W. F. Lages, "Control of a brachiating robot for inspection of aerial power lines," *Applied Robotics for the Power Industry (CARPI), 2010 1st International Conference on*, Montreal, QC, 2010, pp. 1-6.

[11] Fukuda, Toshio, Tadayoshi Aoyama, Yasuhisa Hasegawa, Kosuke Sekiyama, and Shigetaka Kojima. "PDAC-Based Control for the Multi-Locomotion Robot." (2009): n. pag. Web.

[12] Askeland, Stian Hjellvik. "Stabilization of Brachiation Locomotion in a Monkey Robot." Thesis. Norwegian University of Science and Technology, 2012. *Stabilization of Brachiation Locomotion in a Monkey Robot*. June 2012. Web.

[13] Tedrake, Russ. *Drake: A Planning, Control, and Analysis Toolbox for Nonlinear Dynamical Systems"* Computer software. Vers. 0.9.11. 2014. Web.

[14] Tedrake, R. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832).* Downloaded on 3/20/16 from http://underactuated.mit.edu/

[15] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *J Guidance*, 10(4):338-342, July-August 1987.

[16] Gede, Gilbert. "The Direct Collocation Method for Optimal Control." UC Davis. 26 May 2011. Lecture.

[17] Gill, Philip E., Elizabeth Wong, and Michael Saunders. *SNOPT*. Computer software. Vers. 7.5. Web.