

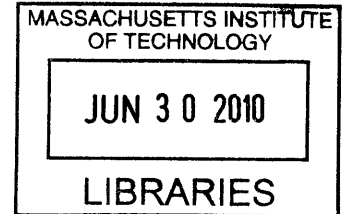
Control of a Nonlinear Underactuated System with Adaptation, Numerical Stability Verification, and the use of the LQR Trees Algorithm

by
Ian Charles Rust

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

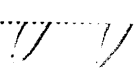
Bachelor of Science
at the
Massachusetts Institute of Technology
June 2010

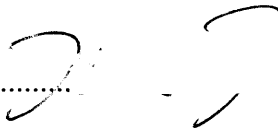
ARCHIVES




© 2010 Massachusetts Institute of Technology
All rights reserved

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Signature of Author.....
 Department of Mechanical Engineering
May 10, 2010

Certified by
 Jean-Jacques E. Slotine
Professor of Mechanical Engineering, Professor of Brain and Cognitive Sciences
Thesis Supervisor

Accepted by
 John H. Lienhard V
Collins Professor of Mechanical Engineering
Chairman, Undergraduate Thesis Committee

Control of a Nonlinear Underactuated System with Adaptation, Numerical Stability Verification, and the use of the LQR Trees Algorithm

by

Ian Rust

Submitted to the Department of Mechanical Engineering
on May 10, 2010 in Partial Fulfillment of the
Requirements for the Degree of Bachelors of Science in
Mechanical Engineering

ABSTRACT

Underactuated robotics, though surrounded by an established body of work, has certain limitations when nonlinear adaptive control principles are applied. This thesis applies a nonlinear adaptive controller that avoids many of these limitations using alterations inspired by the control of a similar underactuated system, the cart-pole. Due to the complexity of the system, a sums-of-squares MATLAB toolbox is used to generate a suitable Lyapunov Candidate used for proofs of stability, with claims of local stability made using Barbalat's Lemma. This provides us with a local domain of attraction for the altered classical nonlinear adaptive controller. In addition, the algorithm known as LQR Trees is applied to the system in order to create a controller with a larger region of attraction and lower torque requirements, though without an adaptive component. Both control systems are implemented in simulations using MATLAB.

Thesis Supervisor: Jean-Jacques E. Slotine

Title: Professor of Mechanical Engineering, Professor of Brain and Cognitive Sciences

Table of Contents

ABSTRACT	2
LIST OF FIGURES	4
INTRODUCTION	6
PROPOSAL OF THE CYLINDER-BEAM SYSTEM	7
BACKGROUND ON ADAPTIVE CONTROL	11
BARBALAT'S LEMMA.....	14
EXPANSION TO THE UNDERACTUATED CASE.....	15
IMPLEMENTATION OF ADAPTIVE CONTROL	17
EXPANDING THE ADAPTIVE CONTROLLER FOR UNDERACTUATION.....	18
THE OBSERVER	23
STABILITY OF THE LINEARIZED SYSTEM	25
REGION OF ATTRACTION	28
LQR TREES	36
GROWING THE TREE.....	39
THE TIME-VARYING LQR CONTROLLER.....	42
REGION OF ATTRACTION OF A BRANCH.....	44
APPLICATION OF THE CONTROLLER.....	46
CONCLUSION	49
APPENDICES	51
REFERENCES	54
ACKNOWLEDGMENT	55

List of Figures

FIGURE 1 – A BASIC REPRESENTATION OF THE CONTROLLER DESIGNED FOR IN THE LQR TREES ALGORITHM. 7	7
FIGURE 2 – A BASIC SCHEMATIC OF THE ROBOTIC SYSTEM PROPOSED FOR THE PURPOSE OF STUDYING UNDERACTUATION, NON-LINEAR CONTROL, ADAPTIVE CONTROL, AND THE LQR TREES ALGORITHM. . 8	8
FIGURE 3 - CHOICE OF GENERALIZED COORDINATES FOR DERIVATION OF THE DYNAMICS OF THE SYSTEM. 9	9
FIGURE 4 - CART-POLE SYSTEM..... 19	19
FIGURE 5 - UNSTABLE FIXED POINT..... 22	22
FIGURE 6 - STABLE FIXED POINT 22	22
FIGURE 7 - RANGE OF C IN WHICH ALL EIGENVALUES OF THE A MATRIX ARE NEGATIVE..... 27	27
FIGURE 8 - STABLE CASE OF THE CONTROLLER. 29	29
FIGURE 9 - UNSTABLE CASE OF THE CONTROLLER..... 30	30
FIGURE 10 – LYAPUNOV CANDIDATE AND ITS TIME DERIVATIVES PLOTTED FOR THE STABLE SIMULATION... 32	32
FIGURE 11 – LYAPUNOV CANDIDATE AND ITS TIME DERIVATIVES PLOTTED FOR THE UNSTABLE SIMULATIO 33	33
FIGURE 12 - REGION OF ATTRACTION BASED ON ANALYSIS OF A LYAPUNOV CANDIDATE FUNCTION AT VARYING INITIAL CONDITIONS FOR α AND $\dot{\alpha}$ 34	34
FIGURE 13 - REGION OF ATTRACTION BASED ON ANALYSIS OF A LYAPUNOV CANDIDATE FUNCTION AT VARYING INITIAL CONDITIONS FOR θ AND $\dot{\theta}$ 35	35
FIGURE 14 - REGION OF ATTRACTION BASED ON ANALYSIS OF A LYAPUNOV CANDIDATE FUNCTION AT VARYING INITIAL CONDITIONS FOR α AND d_0 35	35
FIGURE 15 – REGION OF ATTRACTION OF THE TIME INVARIANT LQR CONTROLLER SEEN IN THE $\alpha, \dot{\alpha}$ PLANE. 38	38
FIGURE 16 - REGION OF ATTRACTION OF THE TIME INVARIANT LQR CONTROLLER SEEN IN THE $\theta, \dot{\theta}$ PLANE.. 39	39
FIGURE 17 - EXAMPLE OF TRAJECTORY OPTIMIZATION FOR RANDOM INITIAL CONDITIONS SEEN IN THE $\alpha, \dot{\alpha}$ PLANE..... 41	41
FIGURE 18 – EXAMPLE OF TRAJECTORY OPTIMIZATION FOR RANDOM INITIAL CONDITIONS SEEN IN THE $\theta, \dot{\theta}$ PLANE..... 42	42

FIGURE 19 – REGION OF ATTRACTION OF CONTROLLER AFTER ADDING 2 BRANCHES, PROJECTED ONTO THE $\alpha, \dot{\alpha}$ PLANE. 45

FIGURE 20 - REGION OF ATTRACTION OF CONTROLLER AFTER ADDING 2 BRANCHES, PROJECTED ONTO $\theta, \dot{\theta}$ PLANE. 45

FIGURE 21 – SIMULATED TRAJECTORY STARTING FROM A RANDOM POINT WITHIN THE BOUNDS OF THE TREE, SEEN IN THE $\alpha, \dot{\alpha}$ PLANE. 47

FIGURE 22 - SIMULATED TRAJECTORY STARTING FROM A RANDOM POINT WITHIN THE BOUNDS OF THE TREE, SEEN IN THE $\theta, \dot{\theta}$ PLANE..... 48

FIGURE 23 - CONTROL INPUT FOR THE TRAJECTORY SIMULATION SEEN IN FIGURES 21 AND 22. 48

Introduction

The control of underactuated systems, though often complex, has an established body of work. As a first step, linearizations can be made, which allow for linear control principles to be applied. These sorts of strategies can be effective, though they are limited due to the fact that all linearizations are based on approximations. When applied to the real system, these approximations only serve to limit the effectiveness of a controller and region of stability of the controller.

Lyapunov analysis is particularly useful as it allows one to avoid this need for linearization. The real dynamics can be used, and as such claims on the stability of the real, nonlinear dynamics can be made. For example, a Lyapunov candidate, a function of the state of a system, is defined as:

$$V(x) \tag{1}$$

If this function is positive definite, and its time derivative is negative definite, then the system is locally stable at the origin. If the function has the following property:

$$V(x) \rightarrow \infty \text{ as } x \rightarrow \infty \tag{2}$$

Then the system is globally stable. There is no need for linearization, and a rather strong claim of stability is made [5].

In addition, adaptation follows quite simply as an extension of this stability analysis (with related analysis using Barbalat's Lemma and the Invariant Set Theorem). This is convenient, since, like linearizations, most models are simply approximations. As is discussed in this thesis, canonical Lyapunov-based proofs of stability for adaptive systems have a distinct reliance on full actuation. In addition, viable Lyapunov candidates are often extremely difficult to determine to for complex systems.

This thesis studies a unique underactuated system as a tool in exploring what claims can be made on the stability of an underactuated system. The system has certain properties that make adaptation applicable, and as such is a suitable platform for study.

Numerical tools are used in order to generate a Lyapunov candidate for the system, and this candidate allows us to detail the regions of local stability in state space.

However, the ultimate goal is to have a globally stable controller. Since we know that claims of local stability can be made, this thesis uses a series of local, linear LQR controllers which covers a larger area of the state space than a single local controller. These controllers are connected in a “tree” in which trajectories are moved from one region of stability to the next until they reach some desired point in state space. In this way all trajectories are simply be “funneled” to the desired point.

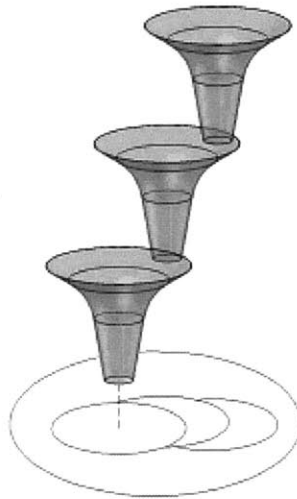


Figure 1 – A basic representation of the controller designed for in the LQR trees algorithm. In the algorithm, the “funnels” are not all stable about a point, but also are stable about trajectories. Notice how a trajectory might cascade from one “funnel” to the next. [8]

The controller applied is simply a function of where in this tree the trajectory is (i.e. the gain matrix is changed depending on where in state space the trajectory is). This is an application of the LQR Trees Algorithm [8]. For simplicity, this task is done without adaptation. Future work may seek to synthesize these two approaches; however that is beyond the scope of this thesis.

Proposal of the Cylinder-Beam System

The underactuated system studied in this thesis is a somewhat complex “toy” robotic system with a biological inspiration. Consider the real situation of a pen balancing

upon a human finger. The finger has a number of inputs which can control the position of the pen on the top of the finger, most of which involve the finger moving in three-dimensional space with six degrees of freedom (translation in three directions and rotation about three axes). The robotic system attempts to emulate this biological balancing system, with certain simplifications. The first is that the system must be constrained to two dimensions. In addition, the control input is reduced to only one control torque, resulting in one rotation. This system thesis called the cylinder-beam system for the remainder of this thesis. It is named as such because it replaces the “finger” with a cylinder forced to rotate about its center and it replaces the “pen” with a beam, forced to only move in two dimensions. This beam is also constrained by a “perfect” frictional contact in which there is no slipping between the beam and ball and they always remain in contact with one another.

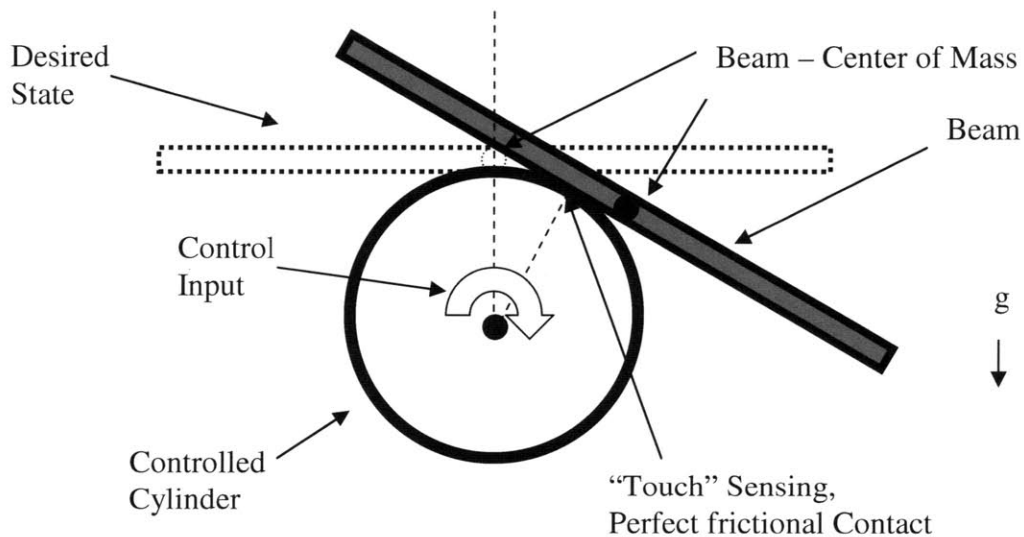


Figure 2 – A basic schematic of the robotic system proposed for the purpose of studying underactuation, non-linear control, adaptive control, and the LQR Trees Algorithm.

This system has several unique properties. The first of these is the fact that there exists “touch” sensing. What this means is that the position and velocity of a contact point between two rigid bodies in the system is measured. In this system the contact point between the beam and cylinder is tracked in this way. The second is the rather unconventional rolling contact between the two bodies, the cylinder and the beam. The beam is able to roll about the cylinder, meaning it is rolling about the contact point.

However, the contact point moves with the beam, making the dynamics somewhat complicated.

In order to derive the dynamics of the system, generalized coordinates must be decided upon. For this system, two angles, θ and α , are able to adequately describe the state of the system (along with the time derivatives of these angles). This gives us four states for this clearly fourth-order system

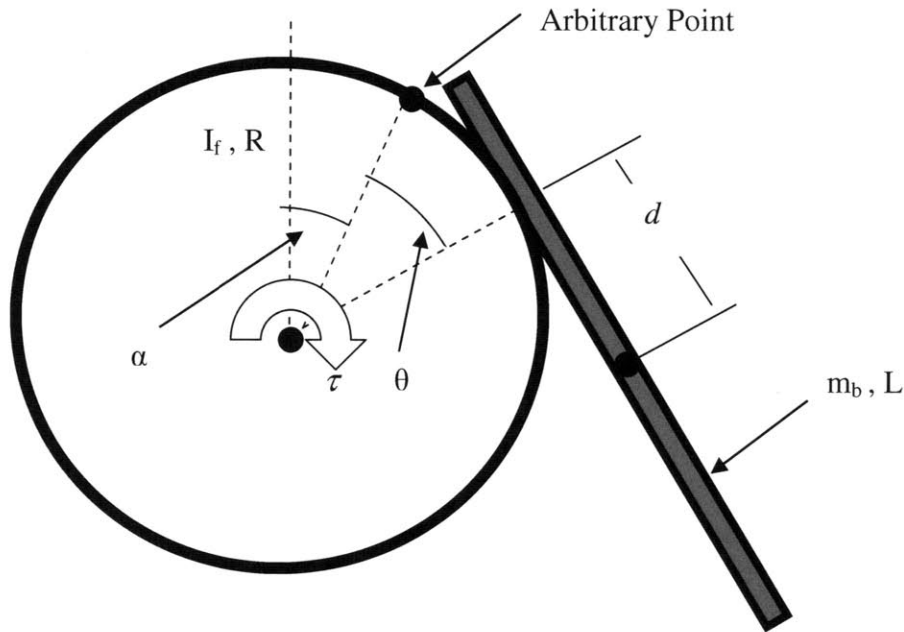


Figure 3 - Choice of generalized coordinates for derivation of the dynamics of the system through Lagrangian mechanics. The coordinate α is the angle of an arbitrary point on the cylinder to vertical. The coordinate θ is the angle from the contact point to the arbitrary point α is in reference to. For complete explanations of the system parameters shown above, refer to Appendix 1.

As seen in Figure 3, α is defined as the angle between an arbitrary point on the cylinder to vertical and θ is defined as the angle between this arbitrary point and the contact point. Other important parameters of the system include the distance of the center of mass to the contact point, denoted as d , which is a function of θ and a constant length d_0 .

$$d = d_0 + R\theta \quad (3)$$

Once this has been done, the equations of motion can be solved for using the methods of Lagrangian Dynamics. For numerical values of the constant parameters in these equations, refer to Appendix 2.

$$I_2 (\ddot{\alpha} + \ddot{\theta}) + m_b R d (\dot{\theta} - \dot{\alpha})(\dot{\theta} + \dot{\alpha}) + m_b g d \cos(\alpha + \theta) = 0 \quad (4)$$

$$I_2 (\ddot{\alpha} + \ddot{\theta}) + I_1 \ddot{\alpha} + 2m_b R d (\dot{\theta} + \dot{\alpha})\dot{\theta} + m_b g (d \cos(\alpha + \theta) - R \sin(\alpha + \theta)) = \tau \quad (5)$$

$$I_1 = I_f + m_b R^2 \quad (6)$$

$$I_2 = m_b \left(\frac{L^2}{12} + d^2 \right) \quad (7)$$

This can be rearranged into the more classical form:

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \begin{bmatrix} 0 \\ \tau \end{bmatrix} \quad (8)$$

$$q = \begin{bmatrix} \theta \\ \alpha \end{bmatrix} \quad (9)$$

$$H(q) = \begin{bmatrix} I_2 & I_2 \\ I_2 & I_1 + I_2 \end{bmatrix} \quad (10)$$

$$C(q, \dot{q}) = m_b d R \begin{bmatrix} \dot{\theta} - \dot{\alpha} & \dot{\theta} - \dot{\alpha} \\ 2\dot{\theta} & 2\dot{\theta} \end{bmatrix} \quad (11)$$

$$G(q) = m_b g \begin{bmatrix} d \cos(\alpha + \theta) \\ d \cos(\alpha + \theta) + R \sin(\alpha + \theta) \end{bmatrix} \quad (12)$$

This verifies the conservation of energy condition [5].

$$\dot{q}^T (\dot{H} - 2C)\dot{q} = 0 \quad (13)$$

The desired state of the system is to have the beam directly on top of the cylinder. What this implies is that the center of mass must be directly vertical and in contact with

the cylinder. Incidentally, these conditions correspond to $\theta + \alpha = 0$ and $d = 0$, respectively. Therefore, the following desired trajectories for the state variables are found:

$$\theta_d = -\frac{d_0}{R} \quad (14)$$

$$\alpha_d = \frac{d_0}{R} \quad (15)$$

$$\dot{\theta}_d = 0 \quad (16)$$

$$\dot{\alpha}_d = 0 \quad (17)$$

One problem with these desired trajectories is that the overall objective of the adaptive control used in the control system presented in this thesis is to stabilize the beam using no prior knowledge about the beam. The constant d_0 is not something that can be directly sensed, and falls under the domain of information about the beam, more specifically the way in which it first comes into contact with the beam. Therefore, as thesis explored in much more depth later in this study, an observer is needed to determine an estimate of the constant d_0 .

The control system applied to this robotic system uses several principles of nonlinear adaptive control. The following sections will detail how these principles are applied to this system, as well as where they must be refined and altered in order to accommodate for various complexities inherent in the system. The most notable of these complexities, of course, is underactuation. In order to fully understand the approach to underactuation, we must look at how adaptive control using a sliding variable controller is implemented and how its stability is verified for a fully actuated system.

Background on Adaptive Control

Adaptive control in fully actuated, nonlinear systems has a well-established body of work [5]. These tools are particularly useful, as stability can almost always be guaranteed through Lyapunov and similar analysis (i.e. the Invariant Set Theorem and

Barbalat's Lemma). Fully actuated systems are common, as are error in parameters such as mass, damping, etc. Without adaptive control, imperfect models would be used, which lends itself to an imperfect controller. Therefore, this particular strategy of control is important and is fortunate to have clear, established proofs of stability. A typical adaptive controller begins with typical autonomous dynamics, written in the general form:

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \quad (18)$$

In order to control this system, a sliding variable must be established. A typical sliding variable is defined as such:

$$s = \dot{\tilde{q}} + \lambda\tilde{q} \quad (19)$$

Where \tilde{q} is defined as the error signal for the controlled variables. Sliding variables are useful because by coercing the variable to a desired value (usually zero), the system simply becomes a first order system defined as such:

$$\dot{\tilde{q}} + \lambda\tilde{q} = 0 \quad (20)$$

This system has clear global exponential stability. Therefore, by controlling the sliding variable, both position and velocity of the system can be controlled. Further definitions using this sliding variable will be particularly useful in establishing the basis for adaptive control in this thesis.

$$s = \dot{q} - \dot{q}_r \quad (21)$$

$$\dot{q}_r = \dot{q}_d - \lambda\tilde{q} \quad (22)$$

In these equations, \dot{q}_r is a reference of sorts for the sliding variable.

Returning to the adaptive controller, the analysis used to show the stability of the controller is similar to Lyapunov Analysis, and involves a Lyapunov Candidate Function:

$$V = \frac{1}{2} (s^T H s + \tilde{a}^T \Gamma^{-1} \tilde{a}) > 0 \quad (23)$$

Where Γ^{-1} is a negative definite matrix and \tilde{a} is the error in the adaptation parameters, defined as $\tilde{a} = \hat{a} - a$. This defines V as a positive definite function, which allows for future claims on stability to be made. The variable a is a constant vector describing the mass and other unknown parameters of the system. This constant vector is chosen such that $H(q)$, $C(q, \dot{q})$, $G(q)$ all have the following linear relationship with a :

$$H(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + G(q) = Y(q, \dot{q}, \ddot{q}_r) a \quad (24)$$

The reason for this becomes more evident once the Lyapunov candidate is differentiated with respect to time:

$$\dot{V} = s^T (\tau - H\ddot{q}_r - C\dot{q}_r - G) + \dot{\tilde{a}}^T \Gamma^{-1} \tilde{a} \quad (25)$$

After substituting equation (24) into (25), this becomes:

$$\dot{V} = s^T (\tau - Y a) + \dot{\tilde{a}}^T \Gamma^{-1} \tilde{a} \quad (26)$$

This definition allows us to choose control and adaptation laws that, in the style of Lyapunov analysis, make this negative definite. The choices for control and adaptations laws are as such:

$$\tau = Y \hat{a} - K_d s \quad (27)$$

$$\dot{\hat{a}} = -\Gamma Y^T s \quad (28)$$

This leads to the following conclusion:

$$\dot{V} = -s^T K_d s \leq 0 \quad (29)$$

This is a negative definite function. Yet, since it is not a function of all states (a has its own dynamics) it cannot be examined with traditional Lyapunov Analysis. However, Barbalat's Lemma will prove a useful tool in proving the stability of the system [5].

Barbalat's Lemma

Barbalat's Lemma is used to prove the local stability of the classical nonlinear adaptive controller detailed previously. In addition, its basic principles are used to show stability in certain regions in the underactuated system studied in this thesis.

Barbalat's Lemma, in its simplest form, states the following:

If:

- The differentiable function $f(t)$ has a finite limit as $t \rightarrow \infty$
- \dot{f} is uniformly continuous

Then $f(t) \rightarrow \infty$ as $t \rightarrow \infty$

Applied to Lyapunov-like functions, the following corollary is made:

If:

- $V(x,t)$ is lower bounded
- $\dot{V}(x,t)$ is negative semi-definite
- $\dot{V}(x,t)$ is uniformly continuous

Then $\dot{V}(x,t) \rightarrow 0$ as $t \rightarrow \infty$.

For the Lyapunov candidate in the previous section already has been shown to satisfy the first and second criteria. For the third criterion, is verifies by examining the closed loop dynamics:

$$H\dot{s} + (C + K_d)s = Y\tilde{a} \quad (30)$$

From this, if s and \tilde{a} are bounded (which comes from equation (29)), \dot{s} is bounded.

This means that \dot{V} is uniformly continuous, since \ddot{V} is defined as:

$$\ddot{V} = -s^T K_d \dot{s} \quad (31)$$

This shows that $\dot{V} \rightarrow \infty$, and therefore $s \rightarrow 0$ as $t \rightarrow \infty$. This in turn guarantees q converges to the specified desired trajectory [5].

Expansion to the Underactuated Case

In the preceding discussion of adaptive control, certain conclusions are consequences of the system being fully actuated. The most important concern is with the definition of the control law, repeated below:

$$\tau = Y\hat{a} - K_d s \quad (32)$$

For a system with $\dim(\tau) = m$, this equality can only be enforced if τ is entirely non-zero entries. However, this criterion directly correlates to the definition of an underactuated system.

The definition is as follows [7]. For a system with following general definition:

$$\ddot{q} = f_1(q, \dot{q}, t) + f_2(q, \dot{q}, t)u \quad (33)$$

The system is under actuated if the following inequality is true:

$$\text{rank}(f_2(q, \dot{q}, t)) < \dim(q) \quad (34)$$

Returning to the adaptive case, an entry of zero directly corresponds to an f_2 with a rank of $m-r$, where r is the number of zero entries in the system. Since $\dim(q)=\dim(\tau)$, a zero in τ means that the system is underactuated. Therefore, if a system is underactuated, the inputs needed for in the control law for the conventional adaptive control can no longer be applied to the system (some of the inputs will always be zero). This basic difference invalidates the control strategy stated in detail in previous sections. The remainder of this thesis is spent finding a viable substitute for this control system.

To begin, Partial Feedback Linearization (PFL) is used on the cylinder-beam system in order to minimize the effects of the dynamics of θ on the dynamics of α [6]. This particular type of partial feedback linearization is called collocated PFL, since α is the variable which is directly affected by the control input. This is done simply by subtracting (4) from (5). This yields the following dynamics for α :

$$I_1 \ddot{\alpha} + m_b d_0 R (\dot{\theta} + \dot{\alpha})^2 + m_b R^2 \theta (\dot{\theta} + \dot{\alpha})^2 - m_b g R \sin(\theta + \alpha) = \tau \quad (35)$$

This use of collocated PFL allows for the dynamics of the system to be reduced to a linear relationship between some dynamical vector Y and a constant vector a .

$$I_1 \ddot{\alpha}_r + m_b d_0 R (\dot{\theta} + \dot{\alpha})^2 + m_b R^2 \theta (\dot{\theta} + \dot{\alpha})^2 - m_b g R \sin(\theta + \alpha) = Y(\alpha, \dot{\alpha}, \theta, \dot{\theta}) a \quad (36)$$

Where Y and a are defined as:

$$Y(\alpha, \dot{\alpha}, \theta, \dot{\theta}) = \begin{bmatrix} \ddot{\alpha}_r & (\dot{\theta} + \dot{\alpha})^2 & \theta (\dot{\theta} + \dot{\alpha})^2 & \sin(\theta + \alpha) \end{bmatrix} \quad (37)$$

$$a = \begin{bmatrix} I_1 & m_b d_0 R & m_b R^2 & m_b g R \end{bmatrix}^T \quad (38)$$

$$\ddot{\alpha}_r = \ddot{\alpha}_d - \lambda \dot{\alpha}_r \quad (39)$$

Of course, non-collocated PFL is another option for simplifying the dynamics of the system for use in adaptation. This would mean that the effects of the dynamics of α on the dynamics of θ are minimized. When this is performed, the following dynamics for θ are found:

$$\beta\ddot{\theta} + (\dot{\theta} + \dot{\alpha})(m_b R d(\dot{\theta} - \dot{\alpha}) - 2\beta m_b R d \dot{\theta}) + m_b g((1 - \beta)d \cos(\theta + \alpha) + \beta R \sin(\theta + \alpha)) = -\beta\tau \quad (40)$$

$$\beta = \frac{I_1 I_2}{I_1 + I_2} \quad (41)$$

This equation again indicates a linear relationship between a dynamic vector and constant vector, though this relationship would be unnecessarily complex to detail in full in this thesis. The collocated PFL found in equation (35) plays an identical role, and is much simpler. Therefore, it is used as the stability characteristics of the controller proposed in this thesis are explored.

Implementation of Adaptive Control

Equation (36), found through collocated PFL, allows for us to design an adaptive controller exceedingly similar to classical controller detailed previously. This begins by choosing a similar Lyapunov candidate function.

$$V = \frac{1}{2}(s_\alpha^T H s_\alpha + \tilde{a}^T \Gamma^{-1} \tilde{a}) > 0 \quad (42)$$

In this the sliding variable is defined in terms of the error $\tilde{\alpha}$.

$$s = \dot{\tilde{\alpha}} + \lambda_\alpha \tilde{\alpha} \quad (43)$$

$$\tilde{\alpha} = \alpha - \alpha_d \quad (44)$$

Where α_d is the desired trajectory for α . As was the case for the classical controller, the adaptation and control laws take on the following form:

$$\tau = Y\hat{a} - K_d s_\alpha \quad (45)$$

$$\dot{\hat{a}} = -\Gamma Y^T s_\alpha \quad (46)$$

Y is the vector of state-dependent dynamic terms detailed in full in equation (37). With these definitions, the derivative of the Lyapunov function can be reduced to a negative definite function.

$$\dot{V} = -s_\alpha^T K_d s_\alpha \leq 0 \quad (47)$$

Through Barbalat's Lemma, in an identical proof to that seen in equations (29) through (31), it is shown that $\dot{V} \rightarrow 0$, and thus α tends towards the desired trajectory.

This is a perfectly valid controller, however it only stabilizes α , and makes no claims on the stability of θ . In order to ensure that θ is also stabilized, additional terms must be added to the control law. The next section details one strategy for doing this, and provides some background on another underactuated system which uses a similar strategy to confront an equally similar problem.

Expanding the Adaptive Controller for Underactuation

The strategy for control of the non-collocated variable θ is exceedingly similar to the control of the collocated variable x seen in the cart-pole system shown in Figure 4. The cart pole is a relatively simple system, though it is underactuated and has distinctly nonlinear dynamics. In it, a force is applied to a cart which is constrained to move along a line, parameterized by x . Attached is a mass that is fixed to rotate about some point on the cart (the "pole"). The overall objective of the control system is to swing the pole to the top of the cart, as well as to place the cart at some point in x (usually $x = 0$)

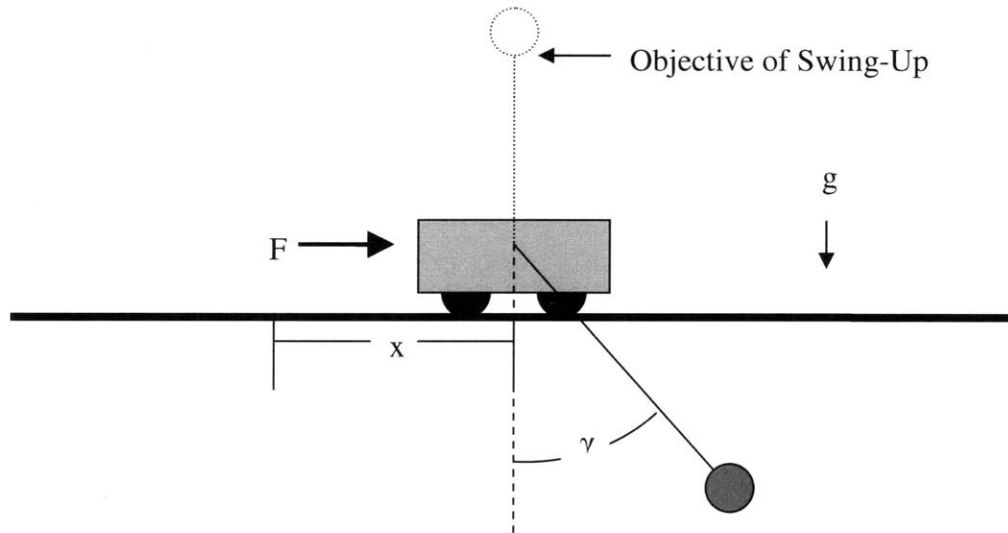


Figure 4 - Cart-Pole System. A control strategy used for this system is adapted to the cylinder-beam system. In this system, a force can be exerting laterally on the cart.

Energy-shaping control is often used in order to swing the pole above the cart. This is an effective tool for this situation, for if the energy of the system always approaches the energy of the system when the pole is directly above the cart and all velocities are zero, only two situations can exist. The first of these is that the pole will eventually be directly above the cart and all velocities are zero. This occurs when the initial energy is anything but the desired energy. The second is that, if the energy is already the desired energy, the system will stay in its current state. This second situation is unrealistic, but in either situation the pole is in the desired location. As such energy shaping is an excellent choice for swing-up control.

As an aside, energy shaping control, as it is seen in the cart-pole system, may seem like a suitable choice for controlling the cylinder-beam system. However, for reasons that are not the focus of this thesis, the cylinder-beam system is unsuited to energy shaping control since the system is not constrained in the vertical direction like the cart-pole. What this means is that the cart pole is unable to move vertically beyond simply rotation of the pole. The beam in the cylinder-beam system is only fixed by a frictional constraint. This means that the potential energy of the system has no lower boundary. As such, it can assume a trajectory that allows the energy to converge to a desired point by simply having fallen off the side of the cylinder, with increasing kinetic energy. The kinetic energy gained in speeding up during the fall is compensated by a loss

of gravitational potential energy. As such, a stable trajectory in terms of energy can be highly unstable in terms of the parameters that the controller would seek to stabilize.

The dynamics of the cart-pole, for the purposes of this thesis, are inconsequential. The most important aspect of the dynamics is the fact that with a suitable choice for a control results in an exceedingly useful property of the energy of the system. Note that this is done using collocated PFL [1].

$$\dot{\tilde{E}} = -K\dot{\gamma}^2 \cos(\gamma)^2 \tilde{E} \quad (48)$$

$$\tilde{E} = E - E_d \quad (49)$$

$$K > 0 \quad (50)$$

As can be seen in these "dynamics" for the energy, the error of the energy, defined by some desired energy E_d , tends towards zero as $t \rightarrow \infty$. Therefore, the pole is swung-up to the vertical position.

However, there are no claims in this controller on the stability of the state x . Therefore, some terms must be added in order to shape the behavior of this state. The obvious choice for this is the simplest. A linear, PD (proportional and derivative) for x is placed in the control input. Therefore, the force input (with unit masses, lengths, etc.) becomes:

$$F = \left[(-\sin \gamma \cos \gamma - \dot{\gamma}^2 \sin \gamma) + (2 - \cos(\gamma)^2)v \right] \quad (51)$$

$$v = k\dot{\gamma} \cos \gamma \tilde{E} - K_1 x - K_2 \dot{x} \quad (52)$$

This system has proven stability, despite the fact that the convergence of the energy seen in equations (29) to (31) is no longer valid. This is shown using Floquet Theory, which begins with linearizing the dynamics of the system about a periodic trajectory. The stability of this is verified by checking the stability of the transverse dynamics along this trajectory [3]. The full details of this particular stability verification are not the topic of this thesis, as the transverse dynamics of the cylinder-beam system are entirely different.

Returning to the cylinder-beam system, this same philosophy is used in order to stabilize θ in the cylinder-beam system. Just like in the cart-pole example, we add PD terms to the controller seen in equation (45) in order to stabilize θ . However, since $\theta = -\frac{d_0}{R}$ is already a stable fixed point (upon examining the dynamics in equation (4)), all that is required is a derivative term. The existence of this fixed point can be shown by first setting all time derivatives in equation (4) equal to zero. When this is done, the following fixed points are found:

$$\alpha + \theta = \frac{\pi}{2}, \frac{3\pi}{2}, \dots, (2n+1)\frac{\pi}{2} \quad (53)$$

$$\theta = -\frac{d_0}{R} \quad (54)$$

However, the first of these is an unstable “fixed” point, while the second is a “stable” fixed point. This is shown by linearizing the system about each of these points. In addition, α is fixed in order to purely show the dynamics of θ . For the situation in equation (53), the general solution of θ indicates instability. For the situation in equation (54), the general solution of θ indicates marginal stability. This can be seen by examining the general solution of these two modes (numerical answers are found using the parameter values in Appendix 2).

$$\theta_{general,unstable} = c_0 0.2009e^{4.8753t} - c_1 0.2009e^{-4.8753t} \quad (55)$$

$$\theta_{general,stable} = (c_0 0.2e^{4.8999it} - c_1 0.2e^{-4.8999it})i \quad (56)$$

For a physical understanding of this, the first fixed point corresponds to the beam perched on left or right edge of the cylinder, with a vertical orientation. This is intuitively unstable (it is much like the vertical point in the cart pole).

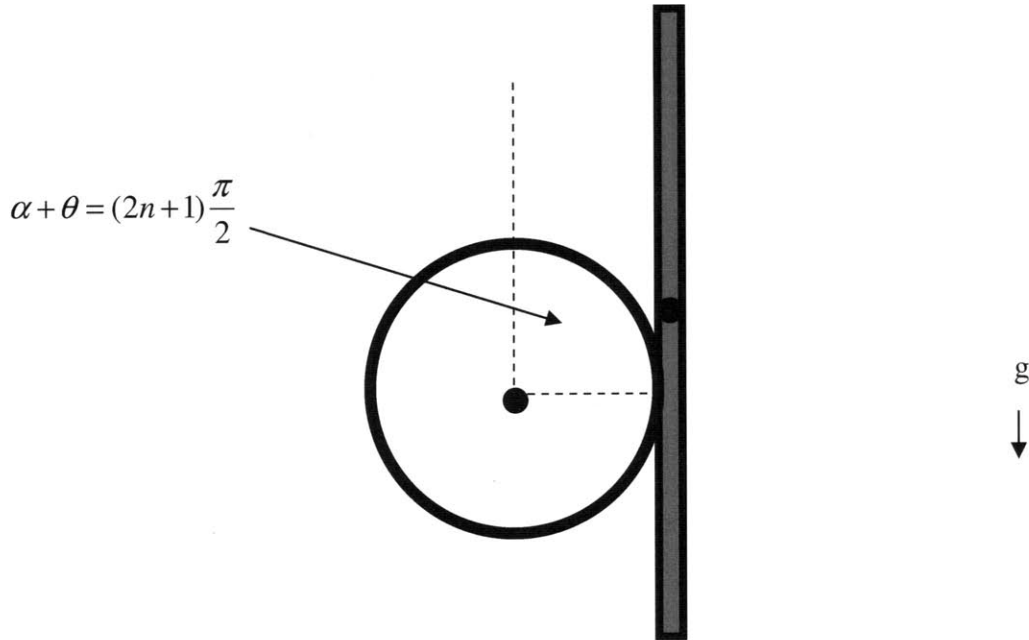


Figure 5 - Unstable fixed point in which the beam is perched on the edge of the cylinder. This situation is intuitively unstable, as it is similar to the vertical position on the cart pole.

The second of these fixed points consists of the beam having its center of mass at the contact point. This is intuitively stable, as then there would no longer be a torque about the center of mass, and as such θ would not vary (though α can).

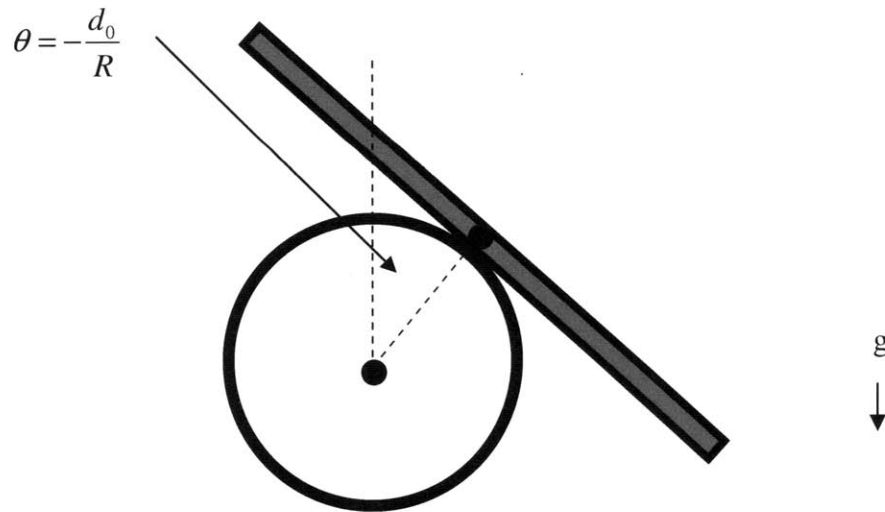


Figure 6 - Stable fixed point in which the beam's center of mass is in contact with the cylinder.

Of course, intuition would also say that additional PD terms would not be detrimental. However, further analysis in this thesis on numerically computing region attraction for this controller has shown that the region shrinks considerably with this term, though the causes are unknown.

Since a proportional term is no longer needed, the control and adaptation laws take on the form:

$$\tau = Y\hat{a} - K_d s_\alpha - cK_d \dot{\hat{\theta}} \quad (57)$$

$$\dot{\hat{a}} = -\Gamma Y^T s_\alpha \quad (58)$$

This, like the cart-pole system, has dubious stability properties once the linear control terms are added. Instead of verifying stability using Floquet Theory, this thesis uses local stability verification using Lyapunov-like analysis with Barbalat's Lemma. In addition, the gain scaling factor c must be tuned using checks for stability. For complete details of the other control constants used in simulation (Γ , K_d , etc.) please refer to Appendix 3.

This system is fairly complex, and as such suitable Lyapunov candidates are exceedingly difficult to find. Fortunately, many numerical tools are available in order to compute Lyapunov functions. The most notable tools are sums-of-squares tools, which are useful because they are able to reduce tests of positive definiteness of a function to a test of positive definiteness of a matrix (something much easier and faster to compute). Future sections of this thesis discuss the implications of the candidate Lyapunov function generated using this method.

The Observer

The goal of the control system designed in this thesis, in a broad sense, is to stabilize the beam on top of the cylinder with no knowledge of the beam altogether (other than the fact that it can be modeled as a two-dimensional "line" of uniform mass distribution). The adaptation implemented in this controller is the first step in achieving this goal. The second and final step toward estimating the parameters of the beam is to

design an observer in order to determine the position of the center of mass of the beam, d_0 .

The necessity of the observer for this constant is apparent in the fact that the desired points for both θ and α are functions of this value. Therefore, if the beam is ever to be stabilized atop the cylinder, this value must be estimated and fed back as a reference for each of these states, θ and α .

Fortunately, there is a rather simple choice for the observer, and has an equally simple explanation of convergence. The observer is designed to have the following dynamics:

$$\dot{\hat{d}}_0 = -\hat{d}_0 - R\theta \quad (59)$$

In essence, this results in \hat{d}_0 being a first-order low-pass filtered version of $-R\theta$, with a cutoff frequency of $1 \text{ rad} / \text{s}$. Upon examining these dynamics, several observations can

be made. The first is that when $\theta = -\frac{\hat{d}_0}{R}$, $\dot{\hat{d}}_0 = 0$. This means that if the controller

detailed previously is able to control θ to this value, the observer converges to some constant value. Therefore, we replace the previous desired trajectories (seen in equations (14) through (17)) with the following desired trajectories:

$$\theta_d = -\frac{\hat{d}_0}{R} \quad (60)$$

$$\alpha_d = \frac{\hat{d}_0}{R} \quad (61)$$

$$\dot{\theta}_d = -\frac{\dot{\hat{d}}_0}{R} = \frac{\hat{d}_0}{R} + \theta \quad (62)$$

$$\dot{\alpha}_d = \frac{\dot{\hat{d}}_0}{R} = -\frac{\hat{d}_0}{R} - \theta \quad (63)$$

Therefore, \hat{d}_0 converges to a constant value if the controller succeeds in stabilizing θ to the desired value detailed above. However, this makes no claim on exactly what constant value to which this observer converges. This constant value can be determined by examining the dynamics (equation (4)). Once again the assumption is made that the controller is stable (at least for some region) meaning that as $t \rightarrow \infty$, α , θ , $\dot{\alpha}$, $\dot{\theta}$ approach the desired trajectories detailed in equations (60) through (63). When this occurs, the dynamics in equation (4) are reduced to a simple equality:

$$d_0 - \hat{d}_0 = 0 \quad (64)$$

This shows that $\hat{d}_0 \rightarrow d_0$ in the case of a stable controller. Therefore, if this observer is implemented, there is a conditional convergence for \hat{d}_0 . If the controller is stable, then $\hat{d}_0 \rightarrow d_0$. If it is not, no further claims can be made. In summary, if stability for the controller can be verified, then the conclusion can be made that the observer converges to the desired value.

Stability of the Linearized System

The first step in stability verification for the controller designed in this thesis is to linearize the system about the fixed point of the system. This point is detailed as being the point at which α and θ each are at their ultimate desired points (i.e. the state at which we seek to stabilize all trajectories towards).

$$\theta_{desired} = -\frac{d_0}{R} \quad (65)$$

$$\alpha_{desired} = \frac{d_0}{R} \quad (66)$$

$$\dot{\theta}_{desired} = 0 \quad (67)$$

$$\dot{\alpha}_{desired} = 0 \quad (68)$$

$$\hat{a} = \text{constant} \quad (69)$$

$$\hat{d}_0 = d_0 \quad (70)$$

Linearization begins by first taking the Jacobian of the nonlinear system, and evaluating it at this desired point. However, since the controller adds four new state variables from the adaptation law and the observer adds one new state variable due to the dynamics of the observer, the order of the system increases. Before the controller and observer were implemented, the system was fourth order. However after the controller and observer are implemented, the system is ninth order. The linearized dynamics of the system therefore can be reduced to:

$$\dot{x} = \left. \frac{\partial f(x)}{\partial x} \right|_{x=x_{desired}} \quad x = Ax \quad (71)$$

$$x = [\theta \quad \alpha \quad \dot{\theta} \quad \dot{\alpha} \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad \hat{d}_0]^T \quad (72)$$

When this is done for the closed loop system, the following A matrix is found:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{2I_1 - m_b g R + (c-1)K_d}{I_1} - \frac{Rg}{L^2} \left(\frac{m_b L^2 + 12I_1}{I_1} \right) & \frac{K_d}{I_1} & \frac{cK_d + I_1}{I_1} & -\frac{K_d + I_1}{I_1} & 0 & 0 & 0 & 0 & -\frac{2I_1 + cK_d}{RI_1} \\ \frac{m_b g R}{I_1} - \frac{2I_1 + m_b g R + (1-c)K_d}{I_1} & -\frac{K_d}{I_1} & -\frac{cK_d + I_1}{I_1} & \frac{K_d + I_1}{I_1} & 0 & 0 & 0 & 0 & \frac{2I_1 + cK_d}{RI_1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -R & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \quad (73)$$

In order to examine the stability of the system, the eigenvalues must be calculated. For this to be done, the system parameters found in Appendix 2 and the control parameters found in Appendix 3 were substituted in. Using $c = 1$, the following eigenvalues are found:

$$\lambda = [9.5564 + 35.2196i \quad 9.5564 - 35.2196i \quad -15.9595 \quad -1.3118 \quad 0 \quad 0 \quad 0 \quad 0 \quad -0.8415] \quad (74)$$

One complex eigenvalue pair is unstable. However, this can be remedied by changing the value of c . There are also four zero eigenvalues. These correspond to the four adaptation states. This merely means that these states will be constants at this point. In addition, the point at which the linearization is centered for the adaptation states does not affect these eigenvalues.

In order to determine the stability as a function of c , the real parts of the eigenvalues were calculated for the values $0 < c < 1$. All values of c that resulted in all negative (or zero) real parts of the eigenvalues were considered acceptable values, as this denotes a stable system. A graphical representation of the results can be found in Figure 7.

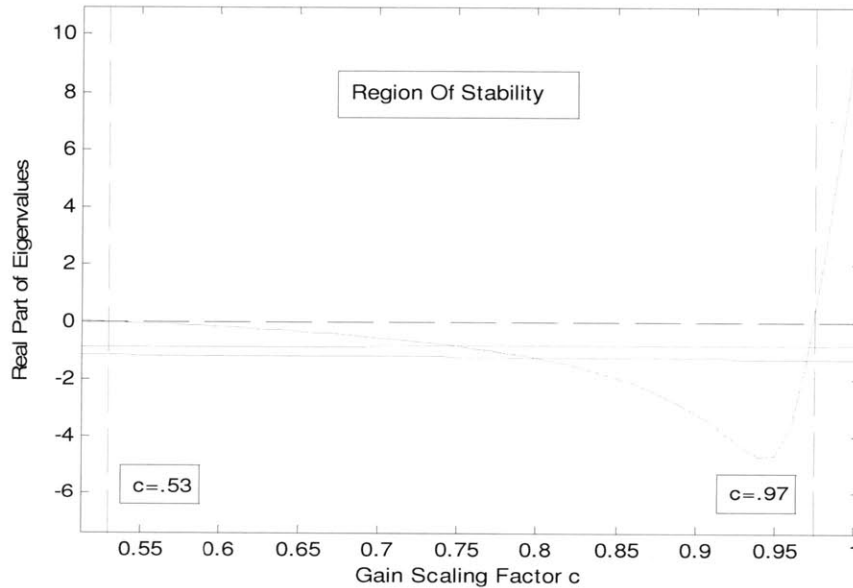


Figure 7 - Range of c in which all eigenvalues of the A matrix are negative (or 0 in the case of the adaptation dynamics). Note that this criteria is primarily dictated by on eigenvalue in particular, and the dashed lines show the transition from stable eigenvalues to unstable eigenvalues.

Upon examining these results, it was found that the range of acceptable values of c (those that resulted in a stable linearized system) was:

$$0.53 < c < 0.97 \quad (75)$$

For the remainder of the study, c is picked to be 0.9. This results in the following stable eigenvalues (real parts are negative):

$$\lambda = [-3.1840 + 15.1475i \quad -3.1840 - 15.1475i \quad -90.2546 \quad -0.8469 \quad 0 \quad 0 \quad 0 \quad 0 \quad -1.2811] \quad (76)$$

Therefore, the linearized system about the desired fixed point is stable, with the exception of the adaptation variables. They are marginally so, but are fixed as constants in this linear domain as seen in the zero entries in the eigenvalues seen above. The following analysis is concerned with discovering when the controller is able to successfully move the system state to the fixed point of this linearization.

Region of Attraction

With the case of fully actuated nonlinear control, a concise proof can be made on the stability of the global stability of the controller. However, underactuation increases the complexity of this proof immensely, and therefore this thesis uses numerical tools in order to explain the stability of the controller. In addition, this controller is not globally stable. There are certain sets of initial conditions that are stabilized, and certain initial conditions that are not stabilized. To show this, the control system was applied to the system in simulation for two sets of initial condition, one unstable, and one stable. For this, the system and controller parameter values found in Appendix 2 and Appendix 3 were used (note especially that K_d was set to be 1×10^3). The first, the stable case, had the following initial conditions:

$$x_i = \left[\frac{\pi}{4} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \right] \quad (77)$$

$$d_0 = 0.1m \quad (78)$$

Note that the initial length, d_0 , while like an initial condition, also characterizes the dynamics. This means that different values of d_0 will yield a different dynamical system,

which can be seen in the existence of this constant in the dynamics. The results of the simulation showed stability:

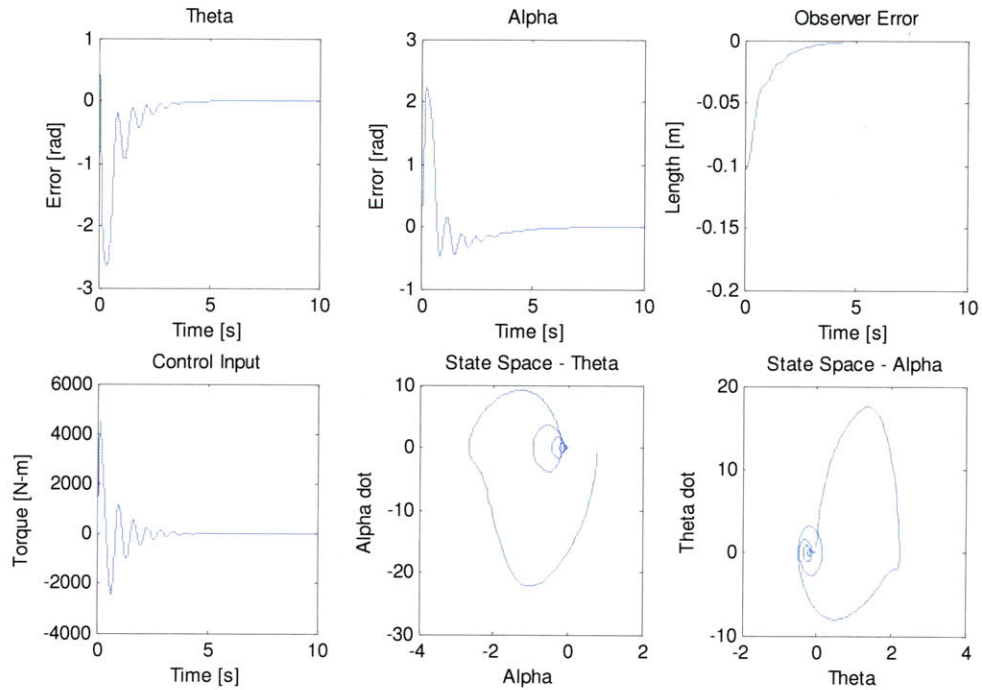


Figure 8 - Stable case of the controller. Position errors are in reference the desired trajectory (a function of the observed parameter d_0).

In comparison, the controller is unstable with slightly different initial conditions:

$$x_i = \begin{bmatrix} \frac{\pi}{4} & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (79)$$

$$d_0 = 0.1m \quad (80)$$

When the exact same controller is acting, and these slightly different initial conditions are applied, the following behavior is seen:

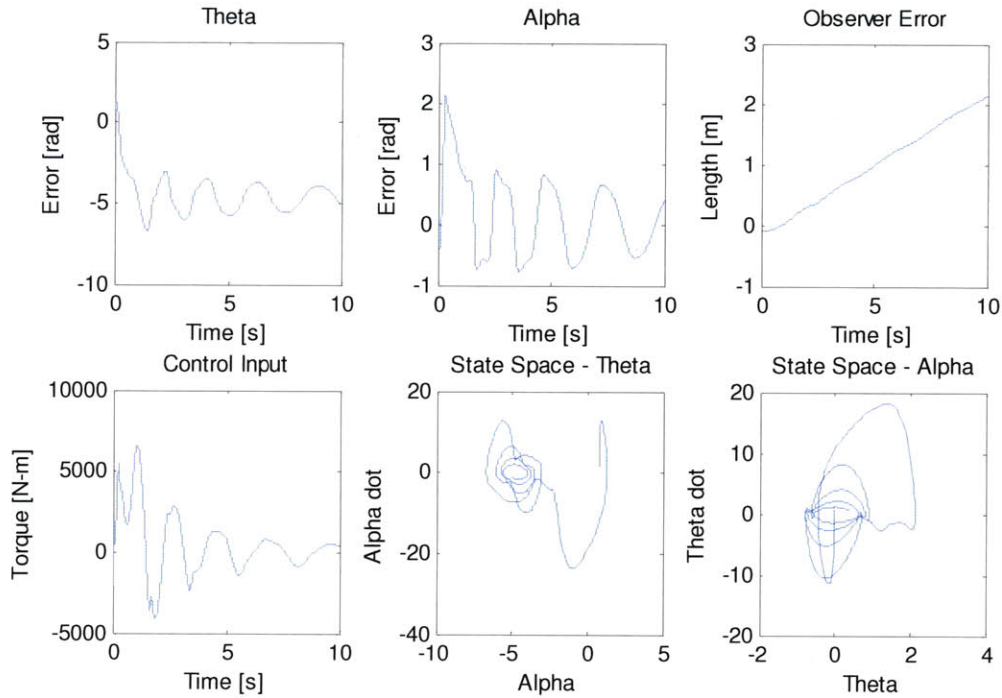


Figure 9 - Unstable case of the controller. Position errors are in reference the desired trajectory (a function of the observed parameter d_0). Note the slight variation of initial conditions as compared to the stable case.

It can be seen from these two cases that the controller is only locally stable, meaning that it only stabilizes the system for a specific set of desired initial condition. As stated previously, computational tools are well suited for this particularly complex case. Sums-of-squares tools are applicable to this problem, as they are able to reduce tests of negative and positive definiteness for functions to tests of positive definiteness of a matrix. For example, take the simple polynomial:

$$1 + 2x^2 \tag{81}$$

By examination, this is obviously a positive definite polynomial. That is not as easily seen computationally. Yet, one observation that can be made is that this polynomial is exactly equivalent to the following:

$$\begin{bmatrix} 1 & x \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = x^T A x \quad (82)$$

This product is always positive definite, if the matrix A is positive definite. This comes directly from the definition of a positive definite matrix.

$$x^T A x > 0 \text{ for any } x \quad (83)$$

In cases of much larger polynomials, this insight allows for rapid tests of positive definiteness in a computational environment. There has been a toolbox written for MATLAB which uses this for various goals. The toolbox is SOSTOOLS, and it is used in this thesis to determine the region of attraction for the controller [4]. This is done by simply solving for a Lyapunov candidate function which would satisfy Barbalat's Lemma for a certain region. This is in turn done by ensuring that the candidate function is negative definite for this region, which is the same as a test for positive definiteness, except the A matrix used is $-A$.

One unfortunate aspect of this sums-of-squares method is that it requires tests of positive definiteness to be done with polynomials. Therefore, the sinusoidal terms in the dynamics cannot be used. Instead, they must be replaced by Taylor expansions about the stable point, $\alpha + \theta = 0$.

$$\sin(\alpha + \theta) = \sum_{n=1}^N \frac{(-1)^n}{(2n+1)!} (\alpha + \theta)^{2n+1} \quad (84)$$

$$\cos(\alpha + \theta) = \sum_{n=1}^N \frac{(-1)^n}{(2n)!} (\alpha + \theta)^{2n} \quad (85)$$

N is chosen to be as large as possible such that computation is not unnecessarily slow ($N = 2$ for the following analysis). This allows for a search to be made for a positive definite function with a time derivative that is negative definite, using the dynamics of the closed loop system. This was first done by doing searching for a function with these properties globally for the whole of state space (for the case of $d_0 = 0$). Obviously, from

our results seen in Figure 8 and Figure 9, this is impossible, and the algorithm halts citing potential "infeasibility" in the problem.

However, it does leave us with the current "best guess" for a function of this nature (simply the last function solved for during the iterations of SOSTOOLS), which will prove useful. The full function can be found in Appendix 4, which uses the system and control parameter values found in Appendices 1 and 3. The meaning of this function can be seen when it is evaluated for the same initial conditions used for the simulations seen in Figure 8 and Figure 9.

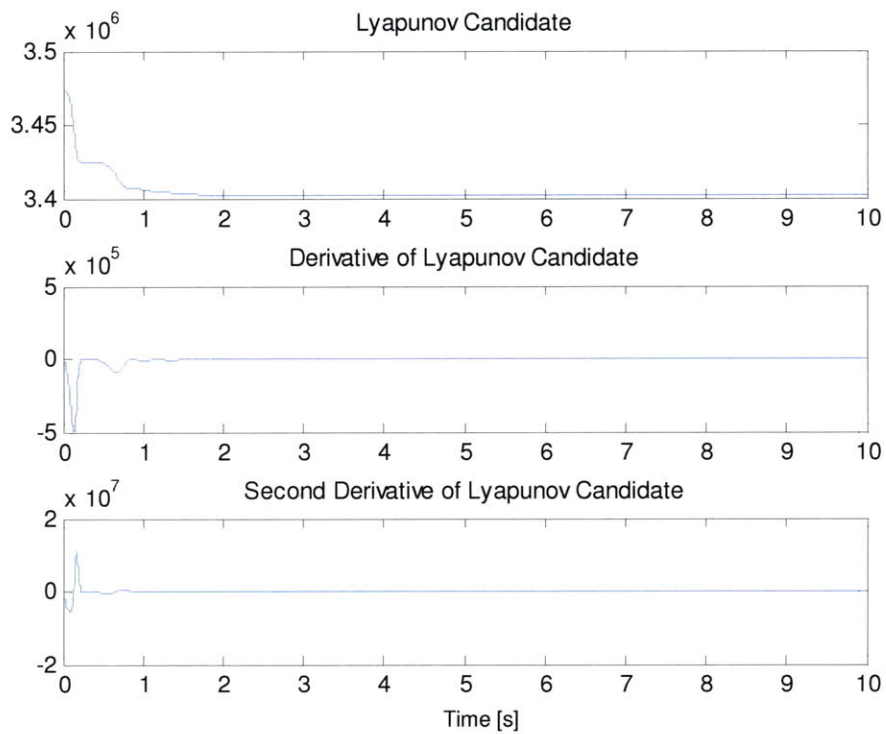


Figure 10 – Lyapunov candidate and its time derivatives plotted for the stable simulation. Note how the first time derivative is always negative, and that $V(x)$ tends towards a finite limit.

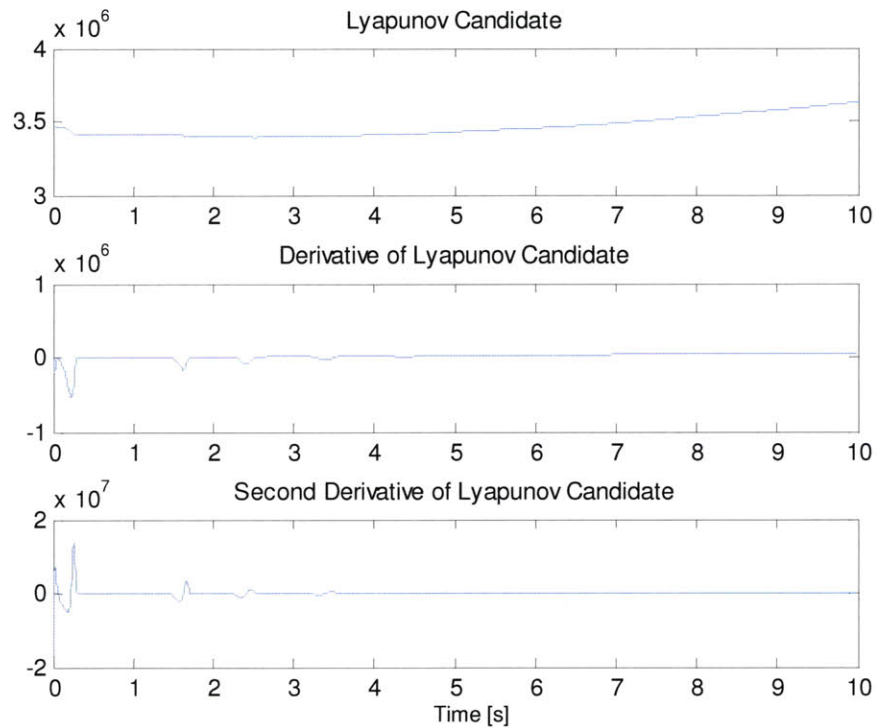


Figure 11 – Lyapunov candidate and its time derivatives plotted for the unstable simulation. Note how the time derivative of $V(x)$ becomes positive at several points (one is marked in the figure).

As is shown in these plots, the stability of the system can be determined by applying Barbalat's lemma to the Lyapunov candidate computed using SOSTOOLS. For the stable case, $V(x)$ is invariably positive definite, as the algorithm searches through only positive definite functions for one with a negative definite time derivative. In addition, since $\dot{V}(x)$ is always less than zero (as shown in Figure 10), $V(x)$ tends towards a finite limit. Lastly, $\dot{V}(x)$ is uniformly continuous, as seen in both the plot of $\ddot{V}(x)$, which is bounded. In addition, this can be seen in the fact that $\dot{V}(x)$ is merely a function of the state, which is bounded because $V(x)$ tends towards a finite limit, and is a function of the state as well. $\ddot{V}(x)$ can be shown to be bounded through this same logic, and as such proves again the uniform continuity of $\dot{V}(x)$. Therefore, $\dot{V}(x) \rightarrow 0$ as $t \rightarrow \infty$. This in turn implies that the state goes to its desired point, since $\dot{V}(x) = 0$ at this point. The equation for $\dot{V}(x)$, shown in full in Appendix 4, verifies that $\dot{V}(x) = 0$ at the desired point. However this is also seen in the fact that $\dot{V}(x)$ is defined as such:

$$\dot{V}(x) = \frac{\partial V}{\partial x} \dot{x} \quad (86)$$

In order for $\dot{V}(x)$ to equal zero, \dot{x} must equal zero. This means that the system will converge to the fixed point, which is the desired state. For the unstable case, $\dot{V}(x)$ does not remain less than zero, and as such the system $\dot{V}(x)$ does not converge to zero.

Therefore, this function is an indicator of stability, as long as $\dot{V}(x) < 0$.

Using this simple metric, the region of attraction of this controller was calculated. By iterating through initial condition, and simulating the system, one can calculate the values of $\dot{V}(x)$ for the simulation. If $\dot{V}(x) < 0$ for this finite time, it can be reasonably concluded that the system is stable. This was done for initial conditions in α , θ , $\dot{\alpha}$, $\dot{\theta}$, as well as for slightly different values for the constant d_0 . The results follow. Note that white regions correspond to the stable regions, and blacks regions correspond to unstable regions.

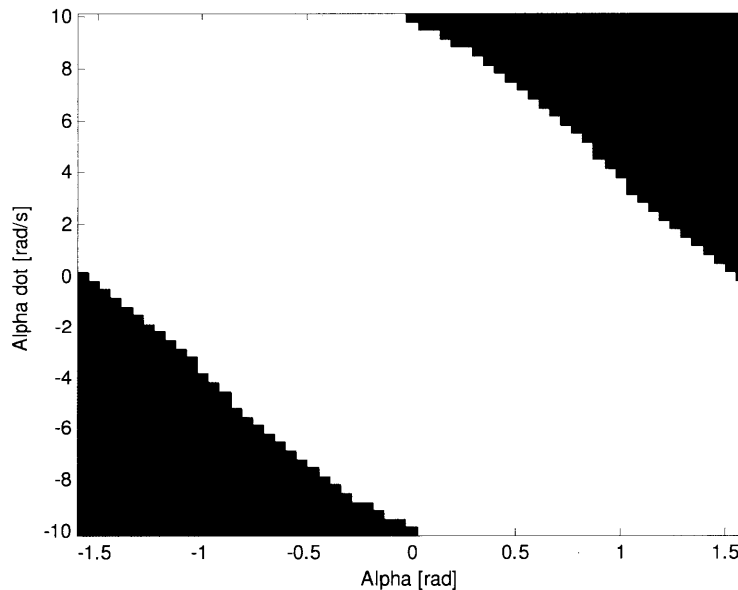


Figure 12 - Region of attraction based on analysis of a Lyapunov candidate function at varying initial conditions for α and $\dot{\alpha}$. Note that the white region corresponds to the attractive region. In addition the bounds on α are the initial conditions in which the beam starts contacting the left and right edges of the cylinder.

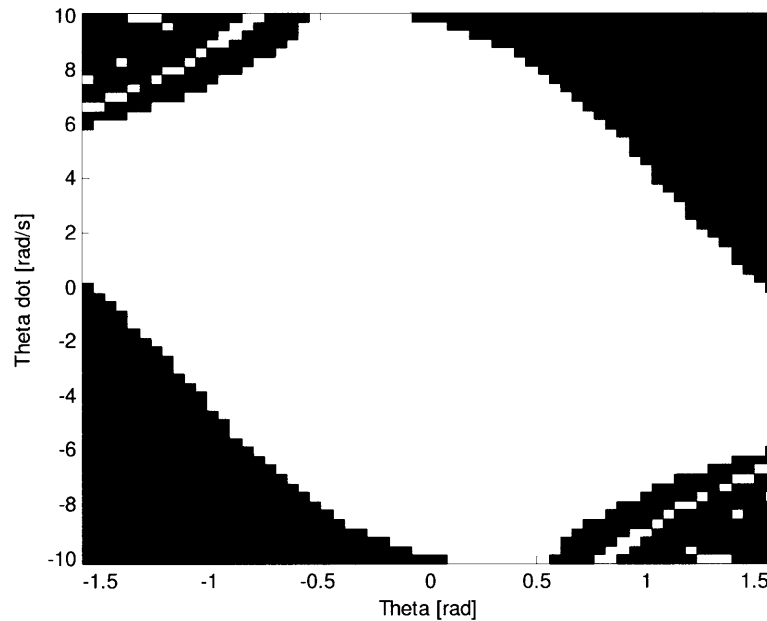


Figure 13 - Region of attraction based on analysis of a Lyapunov candidate function at varying initial conditions for θ and $\dot{\theta}$. Note that the white region corresponds to the attractive region. In addition, the bounds on θ are the initial conditions in which the beam starts contacting the left and right edges of the cylinder.

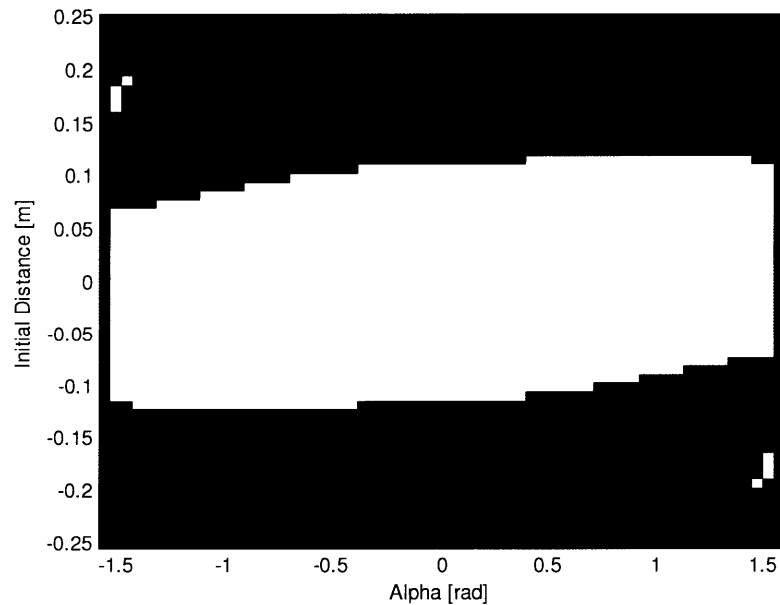


Figure 14 - Region of attraction based on analysis of a Lyapunov candidate function at varying initial conditions for α and d_0 . Note that the white region corresponds to the attractive region. In addition, the bounds on α are the initial conditions in which the beam starts contacting the left and right edges of the cylinder. In addition, be aware that varying values in d_0 must use different Lyapunov candidate function, since the dynamics are changed when d_0 is varied

These regions are "slices" of state space, and the region of attraction has a ninth-dimensional shape. The regions correspond to when the other states are at the desired point, and the states displayed are varied. This allows for this region of attraction to be more plainly seen.

As seen in these figures, the region of attraction is a relatively broad one. Unfortunately, this is done with unrealistically high control inputs on the order of 5000 N-m (see Figure 8). The following section will be concerned with designing a controller that not only reduces the required torque to stabilize the system, but also increases the region of attraction as much as possible.

LQR Trees

The controller detailed previously is able to stabilize the system locally. It has a relatively large region of attraction, yet this can be improved upon. This can be done by applying the LQR Trees algorithm. This algorithm creates a controller that is in actuality a series of locally stable LQR controllers. Each controller is able to coerce trajectories to the region of attraction of the another controller, which passes the trajectory to another region of attraction for another controller, and so on until the trajectory reaches the desired point in state space. The controller exists in a "tree" of these controllers in which any initial point in state space (within a certain range) will follow a chain of controllers to the desired point. For simplicity, no adaptation principles were used when the LQR trees algorithm was applied to the cylinder-beam system.

An additional advantage of using this algorithm is that the linear controllers will require much lower control inputs. The nonlinear adaptive controller has somewhat unrealistic torque requirements (on the order of approximately 5000 N-m). Therefore, another advantage of applying this algorithm is to have much more reasonable torque requirements.

The LQR trees algorithm begins with the controller that is applied last during simulation of the controller. Specifically, this is the LQR controller about the desired point. The tree of controllers terminates at this point, and for this reason it is the "final"

controller. To design the controller, the system was first linearized about the desired fixed point:

$$\dot{x} = \left. \frac{\partial f(x)}{\partial x} \right|_{x=x_{desired}, u=0} x + \left. \frac{\partial f(x)}{\partial u} \right|_{x=x_{desired}, u=0} u = Ax + Bu \quad (87)$$

Using this linearization, the following LQR gain matrix was found using the cost matrices Q and R in Appendix 5 (full state feedback was assumed):

$$K = [-1.9192 \quad 1.6036 \quad -0.8977 \quad 2.1128] \quad (88)$$

In order to determine the region of attraction for this controller, the invariant set theorem can be applied. The invariant set theorem states [5]:

If:

- For $l > 0$, $V(x)$ is bounded in the region Ω_l defined by $V(x) < l$
- $\dot{V}(x) < 0$ for all x in Ω_l

Then $\dot{V}(x) \rightarrow 0$ as $t \rightarrow \infty$.

Therefore, we must find a Lyapunov candidate for which in some region this is true. A suitable candidate is the cost function associated with the LQR controller:

$$V = x^T S x > 0 \quad (89)$$

$$\dot{V} = 2x^T S \dot{x} \quad (90)$$

A region must be found in which the invariant set theorem can be applied. Therefore, a search for the level set l must be done. This can be carried out using an S-procedure, which is done computationally using a sums-of-squares program. The S-procedure

essentially is a search for the maximum l with which the following polynomials are positive definite[8]:

$$-\dot{V} + \lambda(x)(V - l) > 0 \quad (91)$$

$$\lambda(x) > 0 \quad (92)$$

If this is true, then the conditions of the invariant set theorem are met for the region defined by $V(x) < l$ (i.e. $\dot{V}(x) < 0$ in the region). Therefore, an initial condition in this region will be stabilized to the desired point. This computation was done for the controller designed in (88), with the following result:

$$l = 0.6738 \quad (93)$$

This domain of attraction is shown visually in the following figures:

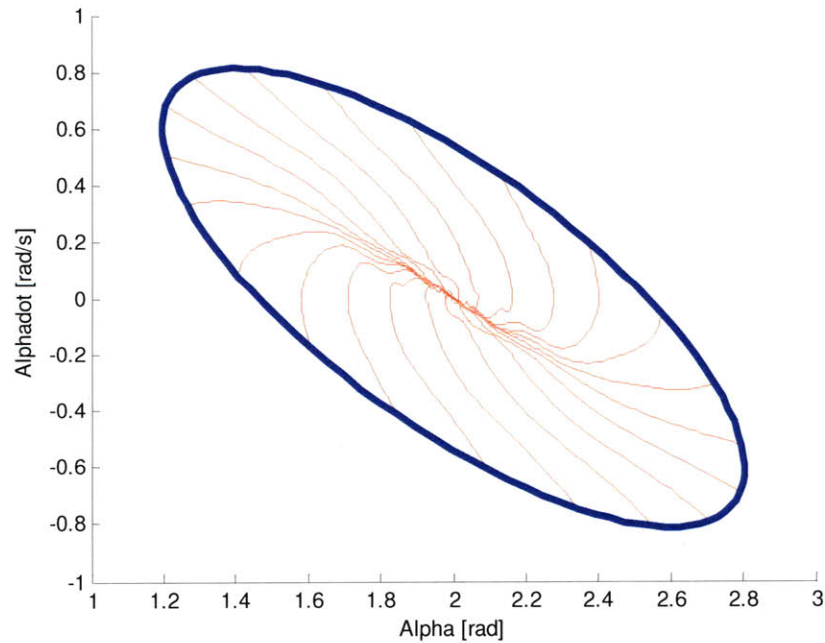


Figure 15 – Region of attraction of the time invariant LQR controller seen in the α , $\dot{\alpha}$ plane. The bold ellipse shows the outline of the region, and the thin lines are sample trajectories initialized on the edge of the region in order to demonstrate the stability of the system in this region.

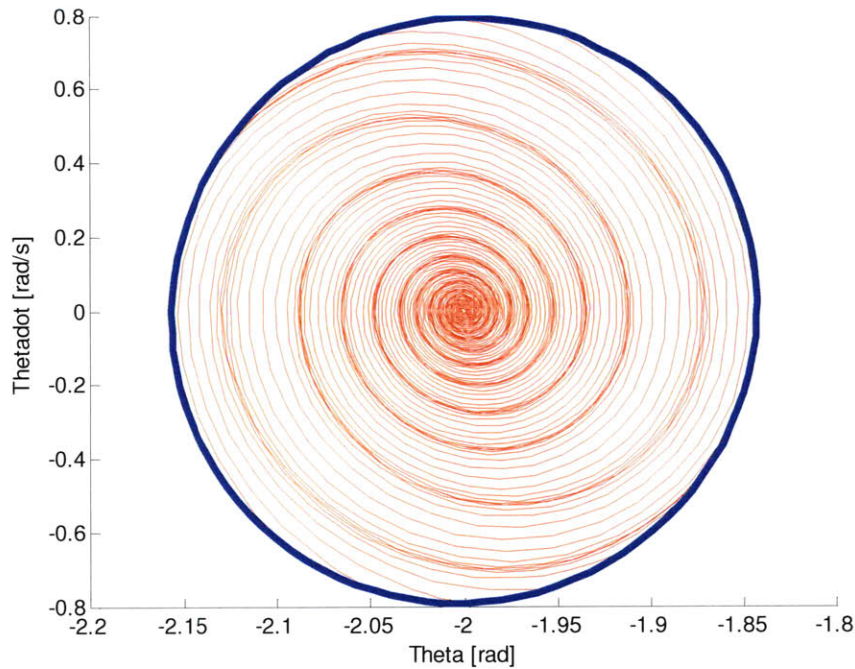


Figure 16 - Region of attraction of the time invariant LQR controller seen in the $\theta, \dot{\theta}$ plane. The bold ellipse shows the outline of the region, and the thin lines are sample trajectories initialized on the edge of the region in order to demonstrate the stability of the system in this region.

This provides us with a region in which this controller can be stably applied.

Growing the Tree

The next step is to create some nominal trajectories leading to this region from random points. Random points are chosen because of the fact that, much like a Rapidly Expanding Random Tree (RRT) algorithm, guarantees can be made stating that as $t \rightarrow \infty$ the entire state space will be filled by the tree [8].

Of course, a choice must be made as to limits of these random points. Because of the physical constraints of the system, this choice is relatively simple. Due to the fact that the dynamics make little sense when the beam is on the bottom of the cylinder, the random points must ensure that the beam begins in contact with the cylinder on some point on top of the cylinder. In addition, θ must be within a range such that the contact point is on the length of the beam. These constraints were tightened slightly in order to

prevent failures in the trajectory optimization algorithm. The final conditions correspond to the following constraints:

$$-\frac{\pi}{2} < \alpha + \theta < \frac{\pi}{2} \quad (94)$$

$$-4 < \theta < 0 \quad (95)$$

The choice of the limits of the initial velocities of the system ($\dot{\alpha}$ and $\dot{\theta}$) is somewhat arbitrary, though the limits chosen were chosen so that the trajectory optimization algorithm used would produce reasonable solutions. The limits are:

$$-1 < \dot{\theta} < 1 \quad (96)$$

$$-1 < \dot{\alpha} < 1 \quad (97)$$

The nominal trajectories from these random points to a point on the existing tree represent new branches in the LQR tree. In order to add new branches to the tree, we must stabilize the trajectories using time-varying LQR controllers, and then determine the regions of stability of the controllers using similar analysis as that for the time invariant case seen above. This allows us to determine exactly which regions in state space are “covered” by the tree. As this “tree” grows, the trajectories may also terminate at points in previous nominal trajectories, depending on what point is closer from a distance metric based on the potential function seen in equation (89). This can be repeated until the regions of stability for all of the controllers combine to cover a suitable large portion of the state space.

In order to generate nominal trajectories, well established shooting methods were used in order to minimize the cost function [1]:

$$J = (x_f - x_{des})^T Q_f (x_f - x_{des}) \quad (98)$$

In this, x_f is the final point in the trajectory, x_{des} is the desired endpoint for the trajectory, and Q_f is a weighting cost matrix (set to be identity). Minimizing this cost function places the end of the trajectory as close as possible to the desired endpoint. This minimization was done using the Sequential Quadratic Programming (SQP) toolbox for MATLAB, SNOPT [2]. Gradients of the cost function were calculated using Back Propagation Through Time (BPTT). An example of this trajectory optimization for a random state can be seen in the following figures:

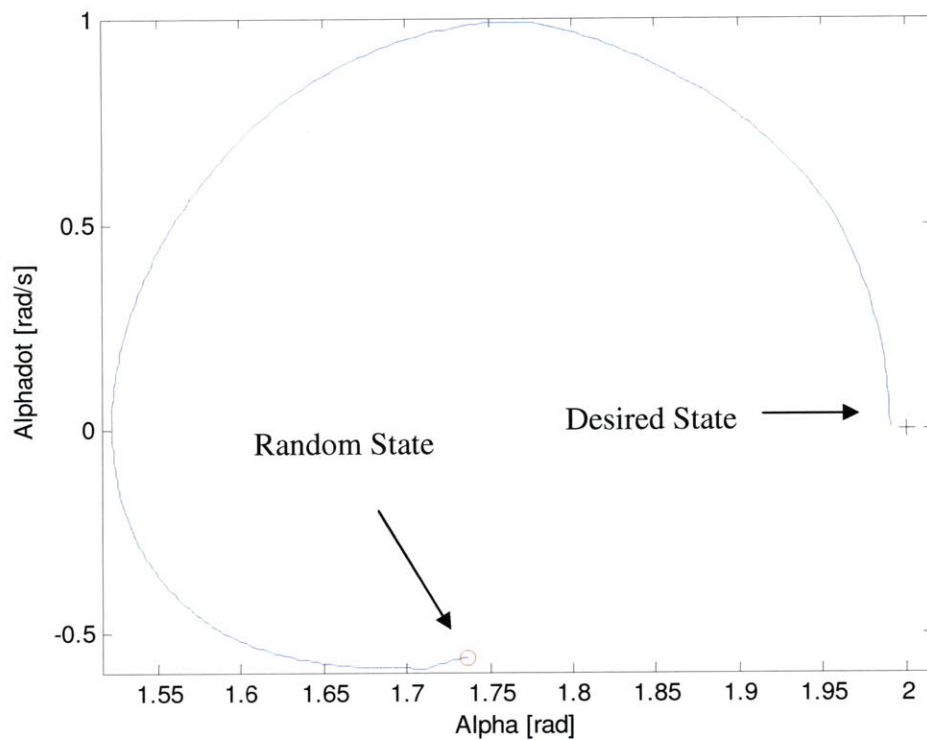


Figure 17 - Example of trajectory optimization for random initial conditions seen in the $\alpha, \dot{\alpha}$ plane

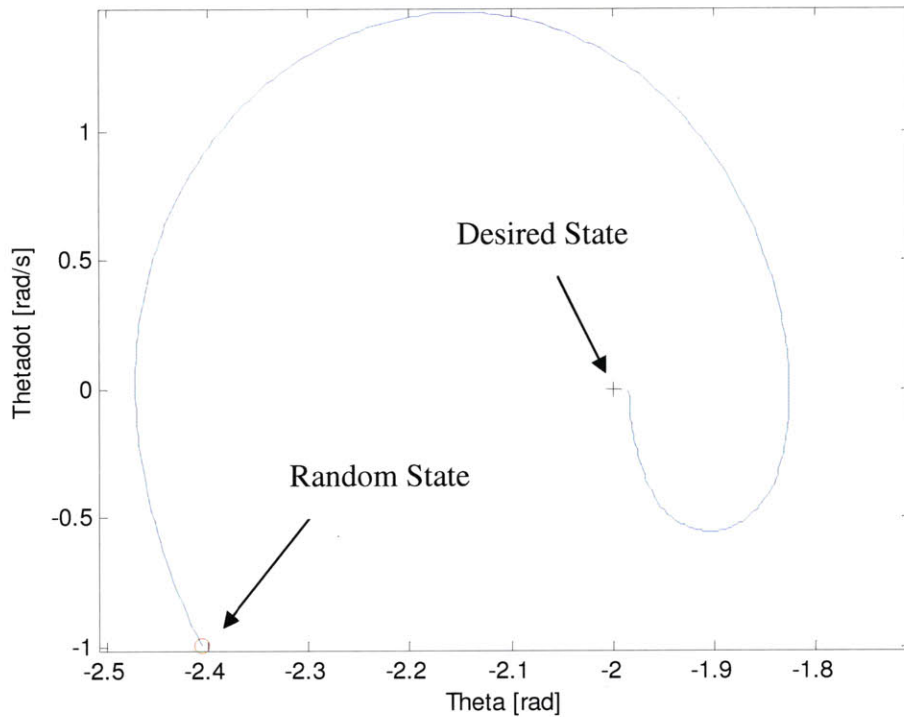


Figure 18 – Example of trajectory optimization for random initial conditions seen in the $\theta, \dot{\theta}$ plane

The Time-Varying LQR Controller

Once a nominal trajectory is found, it must be stabilized using a time-varying LQR controller which coerces nearby trajectories to this trajectory. A time-varying LQR controller can be made by first linearizing the dynamics about the nominal trajectory generated previously:

$$\bar{x}(t) = x(t) - x_0(t) \quad (99)$$

$$\bar{u}(t) = u(t) - u_0(t) \quad (100)$$

$$\dot{\bar{x}} \approx A(t)\bar{x}(t) + B(t)\bar{u}(t) \quad (101)$$

Where $x_0(t)$ and $u_0(t)$ are the nominal trajectories. As with the time invariant LQR controller, a cost function must be used with the form:

$$J = \bar{x}(t_f)^T Q_f \bar{x}(t_f) + \int_0^{t_f} [\bar{x}(t)^T Q \bar{x}(t) + \bar{u}^T R \bar{u}] dt \quad (102)$$

Where Q_f , Q , and R are all positive definite symmetric matrices. By assuming that the optimal cost function takes on the form:

$$J^* = \bar{x}(t)^T S(t) \bar{x}(t) \quad (103)$$

The following Riccati Equation is found, which $S(t)$ is a solution for:

$$\dot{S}(t) = -Q + SBR^{-1}B^T S - SA - A^T S \quad (104)$$

This can be solved by integrating backwards in time, in order to preserve the boundary condition:

$$S(t_f) = Q_f \quad (105)$$

The solution calculated for $S(t)$ leads to the following optimal feedback control law:

$$u^*(t) = -R^{-1}B(t)^T S(t)\bar{x}(t) = -K(t)\bar{x}(t) \quad (106)$$

This feedback law will stabilize trajectories to this nominal trajectory, given the trajectory is initialized within the region of attraction of the controller. The values used for the cost matrices Q , R , and Q_f can be found in Appendix 6. The following section is concerned with determining the limits of this region of attraction for the time-varying case.

Region of Attraction of a Branch

Once a controller is calculated to stabilize a trajectory, the computation of the region of attraction is extremely similar to the procedure for the time-invariant LQR controller. The reason for this is that the trajectory optimization algorithm outputs a discrete array of states that represents the state of the system at each interval Δt in discrete time. As such, the controller is also in discrete time, causing for discrete instances of $S(t_k)$ and $K(t_k)$ for each of the states in the nominal trajectory. As such, the invariant set theorem needs to be applied only to each one of these discrete states, and the corresponding discrete control parameters ($S(t_k)$ and $K(t_k)$). This is done by applying a nearly identical search for a level set l that defines a region $V(x, t_k) < l$ in which $\dot{V}(x, t_k) < 0$ everywhere in that region. Once again an S-procedure is used, which is done by ensuring that the polynomials in (91) and (92) are positive definite. The Lyapunov candidate remains the same. Since the matrix $S(t_k)$ is time-varying, the time derivative of the Lyapunov is slightly different.

$$\dot{V} = 2\bar{x}(t_k)^T S(t_k) \dot{\bar{x}}(t_k) + \bar{x}(t_k)^T \dot{S}(t_k) \bar{x}(t_k) \quad (107)$$

Using this slightly altered S-procedure, a maximum value for l is computed for each state in the nominal trajectory, which results in a varying l in discrete time, $l(t_k)$.

This entire procedure is repeated when a new nominal trajectory is added. The resulting discrete nominal trajectory, defined as $x_{traj}[n]$ and control input $u_{traj}[n]$, and each points corresponding $l[n]$, $S[n]$, and $K[n]$ values, are appended to the values found previously (n is simply an index that allows us to easily relate each point in space to its corresponding level set $l[n]$, $u_{traj}[n]$, $S[n]$, and $K[n]$ values). As more branches are made in this way, the space covered by the regions of attraction defined by $x_{traj}[n]$, $l[n]$, and $S[n]$ expands. A visual representation of the space covered after adding 2 branches can be seen in the following figures.

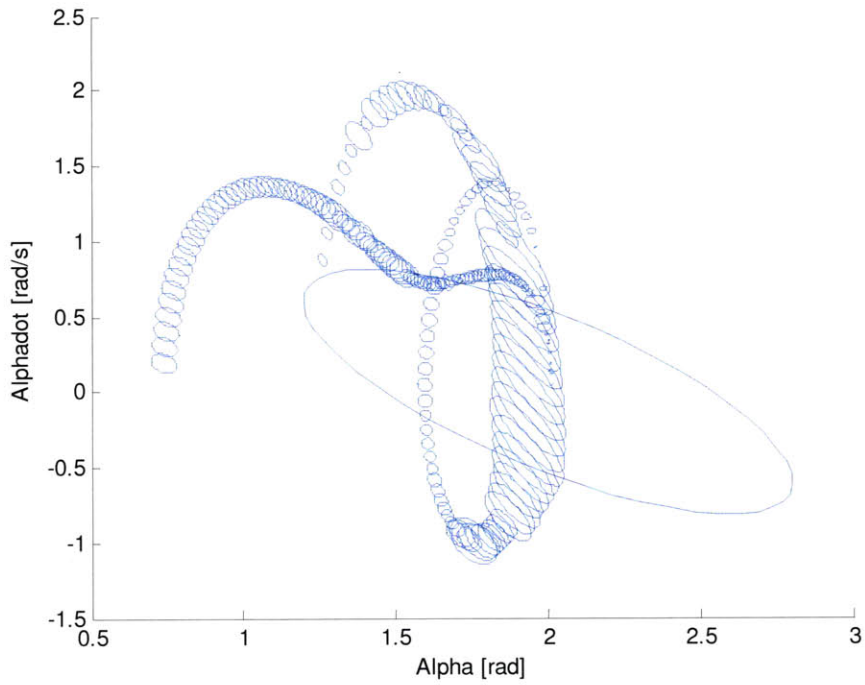


Figure 19 – Region of Attraction of Controller after adding 2 branches, projected onto the α , $\dot{\alpha}$ plane. Note that for the nominal trajectories, the region of attraction (an ellipse) is shown at every third discrete point for clarity.

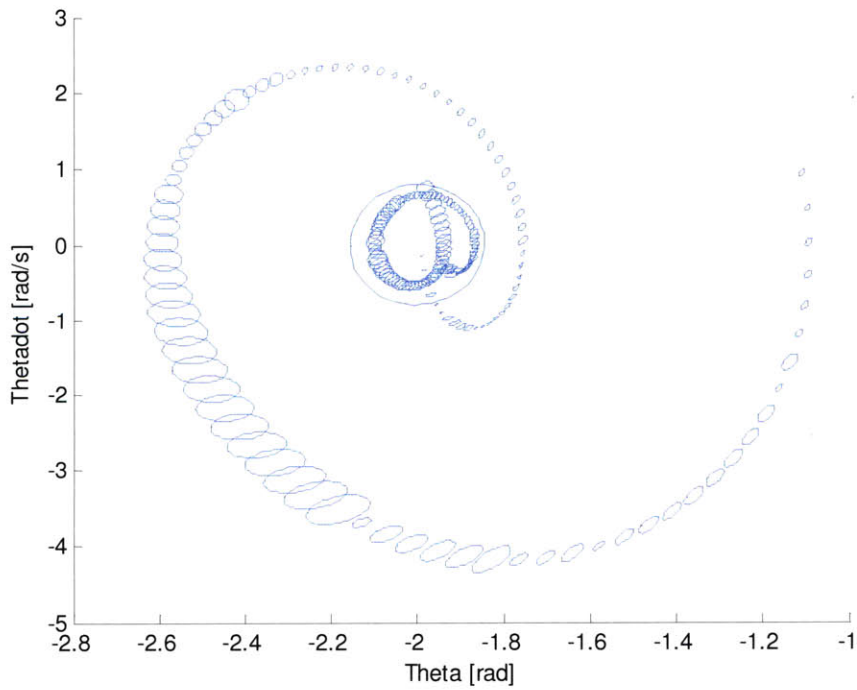


Figure 20 - Region of Attraction of Controller after adding 2 branches, projected onto θ , $\dot{\theta}$ plane. Note that for the nominal trajectories, the region of attraction (an ellipse) is shown at every third discrete point for clarity.

Application of the Controller

Once we have a suitable robust coverage of the state space, we can take the discrete entries of $x_{traj}[n]$, $u_{traj}[n]$, $l[n]$, $S[n]$, and $K[n]$, and apply it to the real system. The algorithm for applying the controller is relatively simple. Firstly, an initial point is randomly selected. Then, the regions of attraction that this point falls into are found. A point is considered inside a region of attraction if, given (108), the inequality in (109) is true:

$$\bar{x}_{init} = x_{random} - x_{traj}[n] \quad (108)$$

$$\bar{x}_{init}^T S[n] \bar{x}_{init} < l[n] \quad (109)$$

Once this is done, any of the regions can be selected, and the corresponding gain matrix $K[n]$ is applied. Any of the applicable (i.e. the initial condition is in the region of attraction) $K[n]$ entries can be applied as such:

$$u_{in} = u_{traj}[n] - K[n](x - x_{traj}[n]) \quad (110)$$

Where x is the current point in state space and u_{in} is the control to be inputted into the system.

For a robust application, at each discrete time step, this procedure is repeated, and an appropriate gain matrix $K[n]$ and nominal control input $u_{traj}[n]$ is applied depending on what region of attraction the trajectory is in. In practice, it is much simpler to merely increment $K[n]$, $u_{traj}[n]$, and $x_{traj}[n]$ at each time step. Note that it is important that the time step used in simulation is the same time step as the trajectory optimization, as inconsistencies in this could cause for the trajectory to “leap” out the region of attraction (for the simulations in this thesis, $\Delta t = .005$). If these time steps are matched, though, this allows for a very simple implementation. Using a controller taken from an LQR tree with

110 branches, a simulation was run starting at some random initial condition within the range of the system defined in (94) and (96). The resulting trajectory is shown.

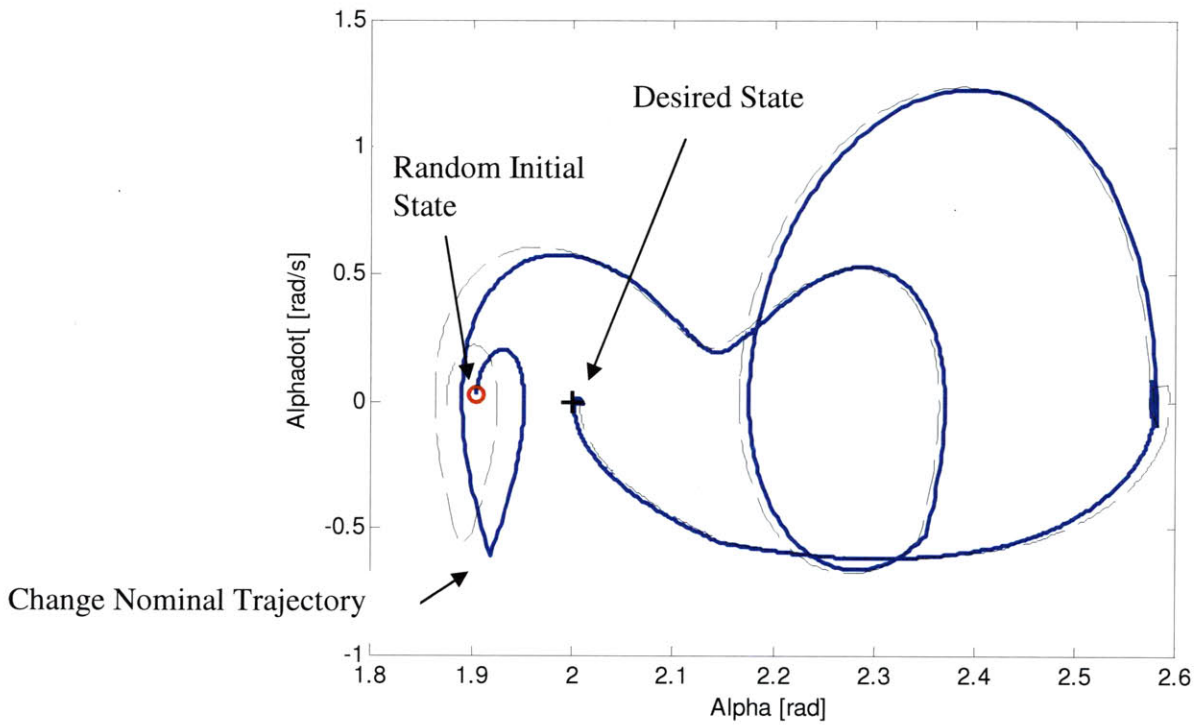


Figure 21 – Simulated trajectory starting from a random point within the bounds of the tree, seen in the α , $\dot{\alpha}$ plane. Note how the trajectory is “passed” from one nominal trajectory stabilization to the next. This occurs at the sharp turns in the trajectory. The nominal trajectory is the dashed line.

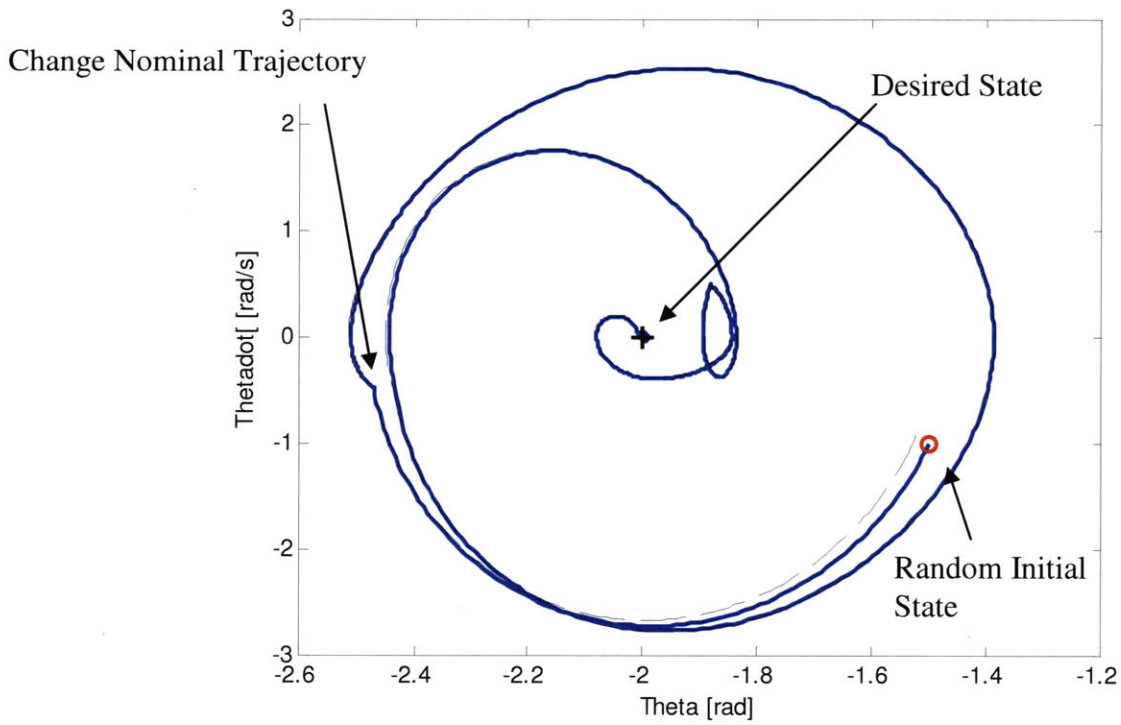


Figure 22 - Simulated trajectory starting from a random point within the bounds of the tree, seen in the $\theta, \dot{\theta}$ plane. Note how the trajectory is “passed” from one nominal trajectory stabilization to the next. This occurs at the sharp turns in the trajectory. The nominal trajectory is the dashed line.

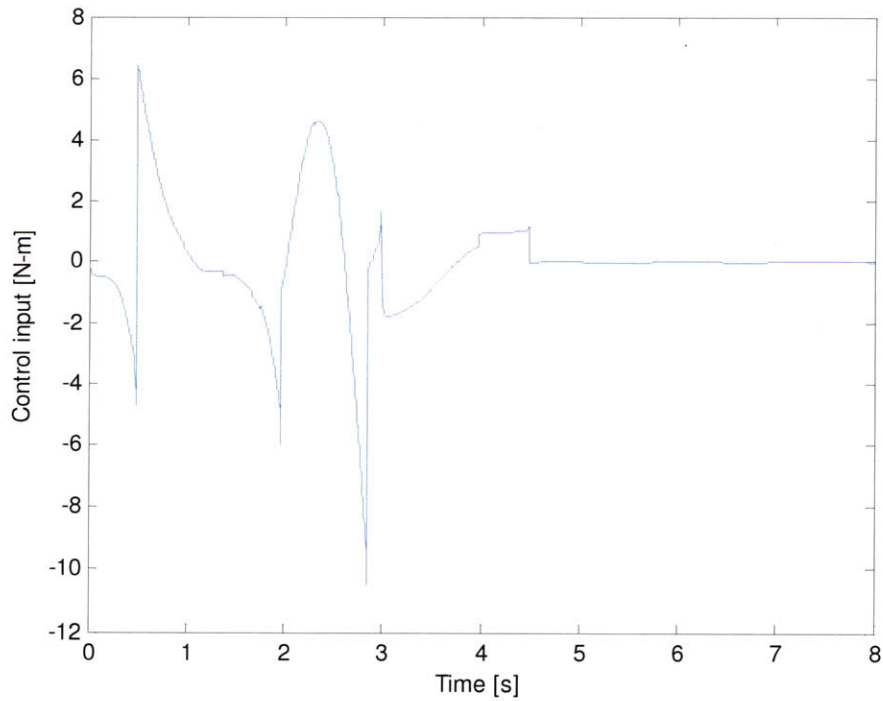


Figure 23 - Control Input for the trajectory simulation seen in Figures 21 and 22.

As can be seen, the trajectory is passed through the branches of the tree, eventually arriving at the desired fixed point. This stability would occur for any initial condition lying within the regions of attraction of the controller. In simulation, the tree was found to cover approximately 80% of the region of state space described by equations (94) through (97). This was calculated by running the controller on a large number of initial conditions, and determining what percentage was stabilized. In addition, the stability is achieved with much lower control inputs, as compared to the nonlinear adaptive controller detailed previously.

Conclusion

This study applied an adaptive controller designed for dynamics discovered using collocated PFL (Partial Feedback Linearization). In addition, using inspiration from work surrounding the cart-pole system, we applied derivative control on the non-collocated variable in order to grant stability to this state along with the collocated states. By using numerical tools, we were able to find a Lyapunov candidate, which although without global applicability, allowed us to determine the region of stability of the controller locally (see Figure 13 through Figure 14).

However, in order to have a controller have a larger region of attraction, the goal of adaptation was ignored, and the algorithm of LQR Trees was applied. This allowed for a local controller (with a larger region of attraction) to be designed for, which was a combination of multiple locally stable controllers. One difference however, is that these were all linear LQR controller, unlike the other adaptive controller, which was nonlinear. One advantage of using these linear controllers, though, is that they required much lower torques in order to stabilize the system (compare Figure 8 to Figure 23).

This study has shown one strategy for nonlinear adaptive control, and explored the stability limits by examining the domain of local stability of the controller. This is a useful strategy, yet the computations for the region of stability are somewhat lengthy. The other controller studied in this thesis, which was generated using the LQR Trees algorithm, was able to cover a somewhat larger area in state space with lower torque inputs, with the sacrifice of adaptive aspects of control. Therefore, the application of each

of these control strategies can be reduced to a decision between range of stability and ability for adaptation. Whatever the outcome of this decision, this thesis has demonstrated the methods and results of each of these control strategies so that they can be applied in the future.

Appendices

Appendix 1: System Parameter Definitions

d_0	Initial distance of contact point from center of mass of bar
g	Acceleration of gravity
m_b	Mass of the beam
I_f	Inertia of the cylinder
L	Length of the beam
R	Radius of the cylinder
τ	Control Input Torque (from motor)

Appendix 2 – System Parameter Values

d_0	$0.1m$
g	$9.8 \frac{m}{s^2}$
m_b	$1kg$
I_f	$1kg \cdot m^2$
L	$0.5m$
R	$0.05m$

Appendix 3 – Control Parameters

Γ	$\begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$
K_d	1×10^3
c	0.9
λ_α	1

Appendix 4 - Full Lyapunov Candidate Function – As Outputted from SOSTOOLS

$$\begin{aligned}
 V(t) = & 0.88614 \cdot a_1^2 + 1387.1508 \cdot a_1 + 0.89512 \cdot a_2^2 + 0.89512 \cdot a_3^2 + \\
 & 0.002631 \cdot a_3 + 0.89552 \cdot a_4^2 - 95.5142 \cdot a_4 + 116.9974 \cdot \alpha^2 + \\
 & 0.88866 \cdot \alpha \cdot \alpha_{\dot{}} + 91.6767 \cdot \alpha \cdot \theta + 0.44244 \cdot \alpha_{\dot{}}^2 + \\
 & 3.5314 \cdot \alpha_{\dot{}} \cdot \theta + 49586.223 \cdot d_{\hat{0}}^2 + 1070.8251 \cdot d_{\hat{0}} \cdot \theta - \\
 & 0.00053733 \cdot d_{\hat{0}} \cdot \theta_{\dot{}} + 44.8339 \cdot \theta^2 + 2.6458 \cdot \theta \cdot \theta_{\dot{}} - \\
 & 0.0064686 \cdot \theta_{\dot{}}^2 + 3473456.2171
 \end{aligned}$$

$$\begin{aligned}
 \dot{V}(t) = & \theta_{\dot{}} \cdot (6451174249352449/70368744177664 \cdot \alpha + \\
 & 17657/5000 \cdot \alpha_{\dot{}} + 588692324382199/549755813888 \cdot d_{\hat{0}} + \\
 & 13229/5000 \cdot \theta_{\dot{}} + 1577452619793485/17592186044416 \cdot \theta) + \\
 & \alpha_{\dot{}} \cdot (4116480055025913/17592186044416 \cdot \alpha + 44433/50000 \cdot \alpha_{\dot{}} + \\
 & 6451174249352449/70368744177664 \cdot \theta) - (44433/50000 \cdot \alpha + \\
 & 11061/12500 \cdot \alpha_{\dot{}} + 17657/5000 \cdot \theta) \cdot (4000000/4001 \cdot \alpha - 4000/4001 \cdot a_4 \\
 & + 4000000/4001 \cdot \alpha_{\dot{}} - 8000000/4001 \cdot d_{\hat{0}} + 4000000/4001 \cdot \theta_{\dot{}} - \\
 & 4000/4001 \cdot a_2 \cdot (\alpha_{\dot{}} + \theta_{\dot{}})^2 + 2/4001 \cdot \theta_{\dot{}}^2 \cdot \theta - \\
 & 4000/4001 \cdot a_1 \cdot (\alpha_{\dot{}} + 40 \cdot d_{\hat{0}} - \theta_{\dot{}} + 2 \cdot \theta) + \\
 & 2/4001 \cdot \alpha_{\dot{}} \cdot \theta_{\dot{}} \cdot \theta - 4000/4001 \cdot a_3 \cdot \theta \cdot (\alpha_{\dot{}} + \theta_{\dot{}})^2 \\
 & + 1/4001 \cdot \alpha_{\dot{}} \cdot \theta \cdot (\alpha_{\dot{}} - \theta_{\dot{}}) + \\
 & 1/4001 \cdot \theta_{\dot{}} \cdot \theta \cdot (\alpha_{\dot{}} - \theta_{\dot{}}) - 196/4001) - (d_{\hat{0}} + \\
 & 1/20 \cdot \theta) \cdot (212971206117163/2147483648 \cdot d_{\hat{0}} - \\
 & 4955994496563177/9223372036854775808 \cdot \theta_{\dot{}} + \\
 & 588692324382199/549755813888 \cdot \theta) + \\
 & (4955994496563177/9223372036854775808 \cdot d_{\hat{0}} + \\
 & 3728894022349925/288230376151711744 \cdot \theta_{\dot{}} - \\
 & 13229/5000 \cdot \theta) \cdot (4000/4001 \cdot a_4 - 4000000/4001 \cdot \alpha - 4000000/4001 \cdot \alpha_{\dot{}} \\
 & + 8000000/4001 \cdot d_{\hat{0}} - 4000000/4001 \cdot \theta_{\dot{}} + 588/25 \cdot \theta \cdot (\alpha + \theta) + \\
 & 4000/4001 \cdot a_2 \cdot (\alpha_{\dot{}} + \theta_{\dot{}})^2 - 2/4001 \cdot \theta_{\dot{}}^2 \cdot \theta + \\
 & 4000/4001 \cdot a_1 \cdot (\alpha_{\dot{}} + 40 \cdot d_{\hat{0}} - \theta_{\dot{}} + 2 \cdot \theta) - \\
 & 2/4001 \cdot \alpha_{\dot{}} \cdot \theta_{\dot{}} \cdot \theta + 4000/4001 \cdot a_3 \cdot \theta \cdot (\alpha_{\dot{}} + \theta_{\dot{}})^2 \\
 & - 12028/100025 \cdot \alpha_{\dot{}} \cdot \theta \cdot (\alpha_{\dot{}} - \theta_{\dot{}}) - \\
 & 12028/100025 \cdot \theta_{\dot{}} \cdot \theta \cdot (\alpha_{\dot{}} - \theta_{\dot{}}) + 196/4001) - (5597/3125 \cdot a_4 - \\
 & 6721214305134235/70368744177664) \cdot (1/2 \cdot \alpha + 1/2 \cdot \alpha_{\dot{}} + 1/2 \cdot \theta) - \\
 & (44307/25000 \cdot a_1 + 3050376868157561/2199023255552) \cdot (\alpha + \alpha_{\dot{}} + \\
 & \theta) \cdot (1/2 \cdot (\alpha_{\dot{}}) + 20 \cdot (d_{\hat{0}}) - 1/2 \cdot (\theta_{\dot{}}) + (\theta)) - \\
 & 11189/12500 \cdot a_2 \cdot ((\alpha_{\dot{}}) + (\theta_{\dot{}}))^2 \cdot (\alpha + \alpha_{\dot{}} + \theta) - \\
 & 1/2 \cdot (\theta) \cdot (11189/6250 \cdot a_3 + \\
 & 6066672957241229/2305843009213693952) \cdot ((\alpha_{\dot{}}) + (\theta_{\dot{}}))^2 \cdot (\alpha + \\
 & \alpha_{\dot{}} + \theta)
 \end{aligned}$$

Appendix 5 – Time Invariant LQR Cost Matrices

Q	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
R	1

Appendix 6 – Time-Varying LQR Cost Matrices

Q	$\begin{bmatrix} 300 & 0 & 0 & 0 \\ 0 & 300 & 0 & 0 \\ 0 & 0 & 300 & 0 \\ 0 & 0 & 0 & 300 \end{bmatrix}$
R	1
Q_f	$\begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 30 \end{bmatrix}$

References

- [1] Fantoni, Isabelle and Rogelio Lozano. *Non-linear Control for Underactuated Mechanical Systems*. Great Britain: Springer, 1973.
- [2] Gill, Philip E. *SNOPT: Software for Large-Scale Linear and Quadratic Programming*. Available from <http://www.cam.ucsd.edu/~peg/>. 2003.
- [3] Hauser, John and Chung Choo Chung. "Nonlinear Control of a Swinging Pendulum." *Automatica* 31 (1995): 851-62.
- [4] Prajna, S., A. Papachristodoulou, P. Seiler, and P. A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*. Available from <http://www.mit.edu/~parrilo/sostools/>. 2004.
- [5] Slotine, J., and Weiping Li. *Applied Nonlinear Control*. New Jersey: Prentice-Hall, Inc., 1991.
- [6] Tedrake, Russ. "The Acrobot and Cart-Pole." 6.832 Underactuated Robotics. Department of Computer Science, MIT Fall 2010.
- [7] Tedrake, Russ. "Fully Actuated vs. Underactuated Systems." 6.832 Underactuated Robotics. Department of Computer Science, MIT Fall 2010.
- [8] Tedrake, Russ, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. "LQR-Trees: Feedback Motion Planning via Sums-of-Squares Verification." *Robotics: Science and Systems Conference*. February 27, 2010.
- [9] Tedrake, Russ. "Trajectory Optimization." 6.832 Underactuated Robotics. Department of Computer Science, MIT Fall 2010.

Acknowledgment

The author would like to thank Professor Jean-Jacques Slotine for his guidance during the research presented in this thesis. In addition, the author would like to thank Professor Russ Tedrake for his help in the implementation of the LQR trees algorithm