Estimation and Control of Flexible Space Structures for Autonomous On-Orbit Assembly

by

Jacob G. Katz

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Department of Aeronautics and Astronautics
May 22, 2009

Certified by .....
David W. Miller
Professor, Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Alvar Saenz-Otero
Research Scientist, Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Prof. David L. Darmofal
Associate Department Head
Chairman, Committee on Graduate Students

Estimation and Control of Flexible Space Structures for Autonomous On-Orbit Assembly

by

Jacob G. Katz

Submitted to the Department of Aeronautics and Astronautics
on May 22, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

The ability to autonomously assemble large structures in space is desirable for the construction of large orbiting solar arrays, interplanetary spacecraft, or space telescopes. One technique uses free-flying satellites to manipulate and connect elements of the structure. Since these elements are often flexible and lack embedded actuators and sensors, the assembly robot must use its own actuators and onboard measurements to suppress vibrations during transportation maneuvers. This thesis will examine the dynamic modeling of a free-flying robot attached to a flexible beam-like element, vision-based estimation of vibrational motion, and trajectory control for assembly of a space structure.

Thesis Supervisor: David W. Miller
Title: Professor, Aeronautics and Astronautics

Thesis Supervisor: Alvar Saenz-Otero
Title: Research Scientist, Aeronautics and Astronautics

Acknowledgments

My sincere thanks to Dr. Alvar Saenz-Otero and Professor David Miller for their advice and guidance throughout my time with the Space Systems Laboratory.

To all my fellow researches in the SSL, it has been a privilege to work with you over the past two years. In particular, I would like to acknowledge Simon Nolet for passing down his vast knowledge of SPHERES, Amer Fejzić and Swati Mohan for their work in developing and completing the second phase of the SWARM project, and Chris Pong for more than once saving SWARM with a last-minute hardware fabrication. Thank you to UROP students Henry Hallam and Fuzhou Hu for significant contributions to the SWARM electronics. A special thanks to Chris Krebs at Aurora Flight Sciences for designing and manufacturing the excellent hardware used in this thesis and for his patience as we worked out the bugs.

To my parents, Ed and Jill Katz, thank you for enriching my life with curiosity and encouraging me to pursue my dreams. Knowing your dedication to education, I am honored to have carried your teachings with me in pursuit of this degree.

Finally, to my friend and companion, Betar Gallant, thank you for walking this road with me, and with your love and support, carrying me through the rough patches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

From the dawn of the space era, satellites and orbiting structures have been constrained to a launch faring or a payload bay. Aside from deploying solar panels or other extendable appendages, satellites today must be designed to fit neatly into the available space. Furthermore, most satellites are launched as a pre-assembled structure that integrates all components for operation. However, even a brief look at the complicated sequence of unfolding petals planned for the deployment of the James Webb Space Telescope[1] suggests that we are reaching the limits of technology for deploying a satellite from a single structure.

Since modern engineering is unlikely to significantly increase available launch volume, any attempts to build large structures in orbit will require better utilization of existing payload space. A promising strategy is to fill the volume with parts of a structure and assemble them after launch. The few missions in space history that have stepped beyond a monolithic structure give a preview of how future space systems might take advantage of orbital assembly. For example, the Apollo program used two separate modules, the Command/Service Module and Lunar Module, to build a spacecraft on orbit capable of landing on the surface of the moon and returning to Earth. Space construction projects, such as Mir or the International Space Station

[1] http://www.nasa.gov/topics/universe/features/jwst_animation.html

(ISS), have been assembled from many individual modules to create a more capable structure. Unfortunately, the success of these projects required colossal efforts of entire space agencies, large expenditures, multiple launches to build and maintain, and human crews for assembly in space. Modern space assembly projects require cost efficient and reliable methods that do not involve the complexities of human operations.

Developing autonomous robots for the assembly of structures in space will provide a key technology for a new generation of large spacecraft. For light-gathering satellites such as space telescopes or orbiting solar arrays, on-orbit assembly provides the attractive opportunity of vastly increasing the surface area available from a single launch vehicle. Instead of relying on a chain of deployment mechanisms, a large array could be assembled by a small helper robot that joins pieces of the structure together. The structures could be assembled over the period of several weeks or months, supervised by a ground team instead of a costly human construction crew.

Due to the use of lightweight materials, construction of large space structures will likely involve the manipulation of flexible elements such as trusses or solar panels. The objective of an assembly robot will be to position and join these components without inducing large vibrations that could disrupt assembly or damage the components. Therefore, it will be critical to consider flexibility in the development of a control and estimation system for autonomous on-orbit assembly.

1.2 On-Orbit Assembly of Flexible Structures

1.2.1 Propulsion Tug Assembly

An orbiting structure might start as an array of parts tightly packed in a palette on a launch vehicle or freely floating after being deposited by several separate launches. From this point, an assembly robot joins the pieces together in their proper configuration. This study considers the assembly of an orbiting space structure with the use of a small robot called a "propulsion tug," such as the vehicle pictured in Figure 1.1.

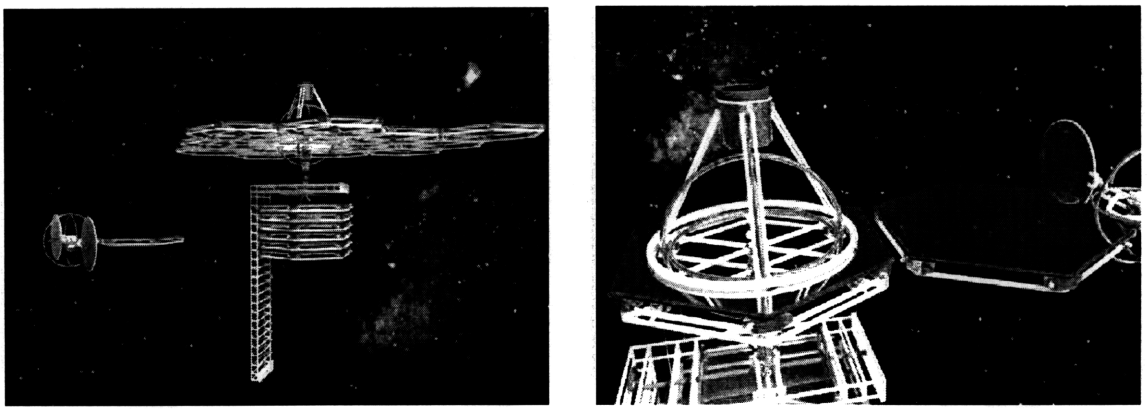


Figure 1.1: A propulsion tug assembles mirror segments of a large space telescope.

The tug adds mobility to passive structural elements by docking to them or grasping them with a manipulator arm. Once attached, the tug uses its thrusters and moment generating devices to maneuver the component into position and join it to the rest of the structure. We will assume from this point forward that a docking mechanism is used to attach the components.

1.2.2 Collocated and Noncollocated Docking

For the purposes of this study we shall consider the maneuvering and docking of a long, flexible beam-like element with a propulsion tug attached at one end. There are two approaches to docking the flexible beam, diagrammed in Figure 1.2, and defined below:

Collocated Docking The docking attachment is located on the same side as the propulsion module. The mechanism may be located above or below the beam, or in some scenarios, the robot may serve as a docking mechanism and become part of the structure after docking.

Noncollocated Docking The docking attachment is located at the far side of the beam. The goal of the propulsion tug is to guide the far end of the beam into the target docking port.

The choice of terms for defining the docking configurations reflects the nature of the control problem for maneuvering the docking mechanism. Collocated control implies

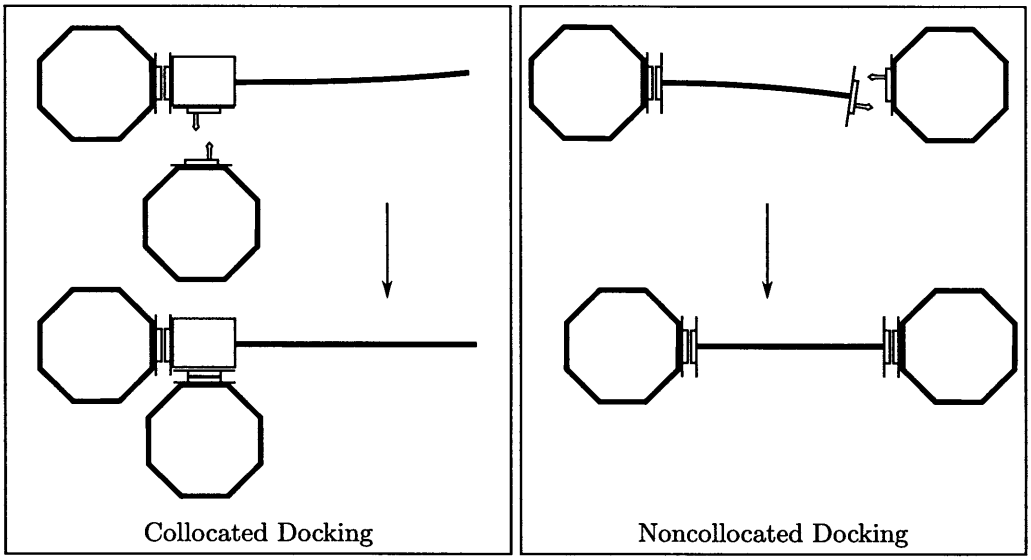


Figure 1.2: In a collocated docking scenario, the propulsion tug is rigidly attached to a docking mechanism, while in the noncollocated case, the docking mechanism is attached to the end of a flexible structure.

that actuators are located in the same location as the outputs or measurements to be regulated, whereas noncollocated control implies that outputs are located elsewhere.

Noncollocated docking is particularly challenging when docking a flexible beam because it requires the manipulation of the free end of the beam using control inputs that are propagated through the flexible dynamics. Instead of attempting to directly steer the end of the beam, we take an alternate approach by recasting the noncollocated docking problem into a collocated one. If a trajectory is selected for the propulsion tug to follow while it simultaneously rejects beam oscillations, we can achieve docking by commanding the propulsion tug directly. The simplest approach to accomplishing this goal is alternating between a transit maneuver that excites the beam dynamics and a damping maneuver that eliminates residual vibrations. Higher level approaches may be added in the form of pre-planned trajectories or control inputs designed to minimize vibrations.

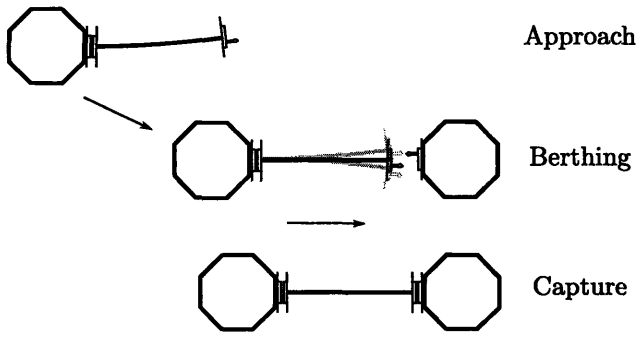


Figure 1.3: To perform beam docking, the tug maneuvers close to the target in the approach phase, damps out beam vibrations in the berthing phase, then completes the docking in the capture phase.

1.2.3 Sequencing of Assembly Events

If vibrations are properly eliminated prior to docking, the set of maneuvers for assembling a flexible structure with a space tug is similar to autonomous docking of two rigid spacecraft [18]. Docking proceeds in approximately three phases, pictured in Figure 1.3:

Approach The tug translates and rotates the flexible element into position at a safe distance from the target. If possible, the element should be positioned to minimize any excitation during the final maneuvers. During this phase, residual vibrations are attenuated before continuing to the next maneuver.

Berthing The tug proceeds to within a few centimeters of docking and executes a final fine alignment with the docking mechanism. Though it may be possible to proceed directly from the end of the approach phase to final docking, this maneuver is performed to eliminate positioning errors and small vibrations induced during the slow approach to the target.

Capture A terminal thrust command is issued to establish a closing velocity between the end of the flexible element and the target docking port. As soon as contact is established, the docking mechanism closes to secure the element to the rest of the structure.

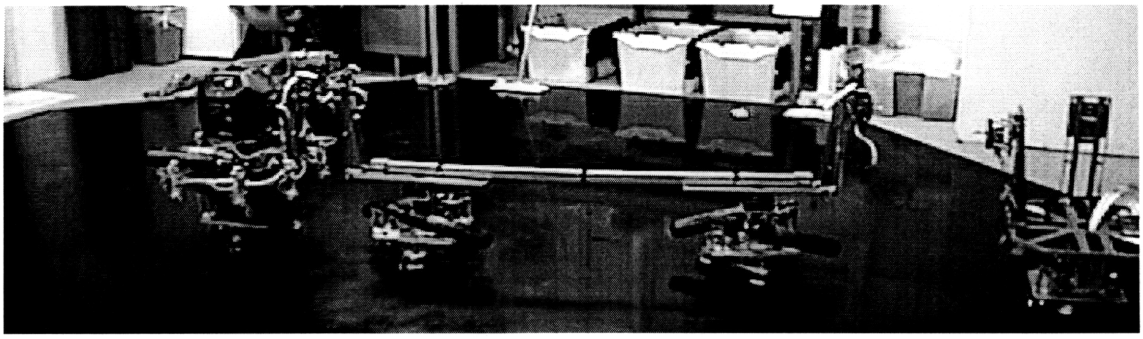


Figure 1.4: The SWARM propulsion module maneuvers a flexible beam element toward a fixed target on the right.

1.3 SWARM and SPHERES Testbeds

Hardware demonstrations for this study were carried out on the Self-Assembling Wireless Autonomous Reconfigurable Modules (SWARM) testbed developed by the MIT Space Systems Laboratory (SSL). SWARM leverages the capabilities of the Synchronized Position Hold Engage Reorient Experimental Satellites (SPHERES) (see [11, 20]) to demonstrate tug-based assembly on a 2D air bearing floor.

1.3.1 SPHERES

The SPHERES program was created to reduce risk in the development of advanced satellite technologies for formation flight, docking, and on-orbit assembly with a low-risk, experiment-driven testing environment. The testbed consists of multiple reprogrammable nanosatellites (4 kg), complete with power, processing, communications, navigation, and propulsion subsystems, and capable of performing full 6 degrees of freedom (6DOF) maneuvers. Ground experiments are performed with air bearing carriages to simulate a frictionless environment in 2D, and flight experiments are conducted in 6DOF on reduced gravity aircraft as well as aboard the International Space Station. For the SWARM project, a satellite is attached to external hardware to augment its capabilities for the task of assembling a structure.

Force per thruster (SPHERES)	0.10 N
Force per thruster (Prop module)	0.10 N
Total force	0.60 N
Total torque	0.10 N·m

Table 1.3.1: SWARM Propulsion Module Thrust Capabilities

1.3.2 SWARM Hardware

Since the project started in 2006, SWARM has had several major hardware revisions, with two major research phases. Phase I investigated the construction of a sparse-aperture space telescope using rigidly connected hardware elements. A SPHERES satellite mounted on an air carriage with two docking ports was used to maneuver and attach components of the structure. Phase II made major hardware improvements to the Phase I components and considered the construction of a structure with flexible elements. The Phase II hardware was used in this study and is summarized below:

Propulsion Module In Phase I, SPHERES thrusters alone were inadequate for overcoming friction while moving an assembled structure on the flat floor. To alleviate this problem, a propulsion module was designed that augments the SPHERES thrusters with 16 additional thrusters located in pairs at each corner of a square mounting plate. Together, the propulsion module and SPHERES satellite simulate the propulsion system of a small assembly tug. The approximate thrust capabilities of the propulsion module are summarized in Table 1.3.1.

Flexible Beam A key objective of the SWARM project is the demonstration of on-orbit assembly in the presence of flexibility, but simulating large flexible structures in a gravitational field is challenging because they are prone to sagging or buckling. To address this issue, the SWARM team designed a beam composed of several rigid links connected by flexible joints, shown in Figure 1.6. The structure is a physical approximation to a homogeneous beam similar to a lumped approximation model in flexible dynamics simulation. Chapter 2 will explore the dynamics of the flexible beam as well as analogies that can be

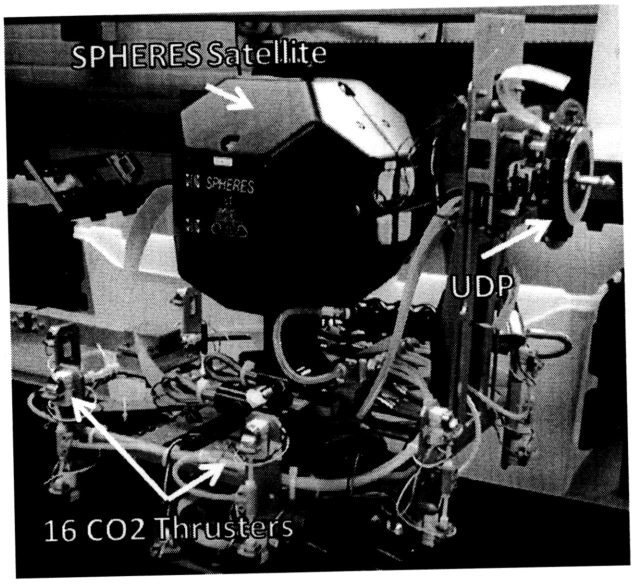


Figure 1.5: The SWARM propulsion module adds additional thrusters and a universal docking port to the SPHERES satellite.

drawn from this model.

Universal Docking Port (UDP) Both the SWARM propulsion module and the ends of the flexible beam are equipped with docking mechanisms by which the elements can be joined together. The UDP is genderless, which allows either end of the beam to be manipulated by or joined to other copies of the docking port. Requirements and specifications for the UDP can be found in [12, 19].

Digital Video Camera The SWARM propulsion module is equipped with a Surveyor Corporation SRV-1 digital video camera. The camera includes an Analog Devices Blackfin digital signal processor for pre-processing camera information before forwarding information via serial port to the satellite. As shown in Figure 1.8, the camera is mounted just above the docking port where it is used to observe the dynamics of the flexible beam.

1.3.3 Global Estimation System

Global position measurements establish the position and orientation of the robot in a pre-defined reference frame. In an on-orbit assembly scenario these measurements

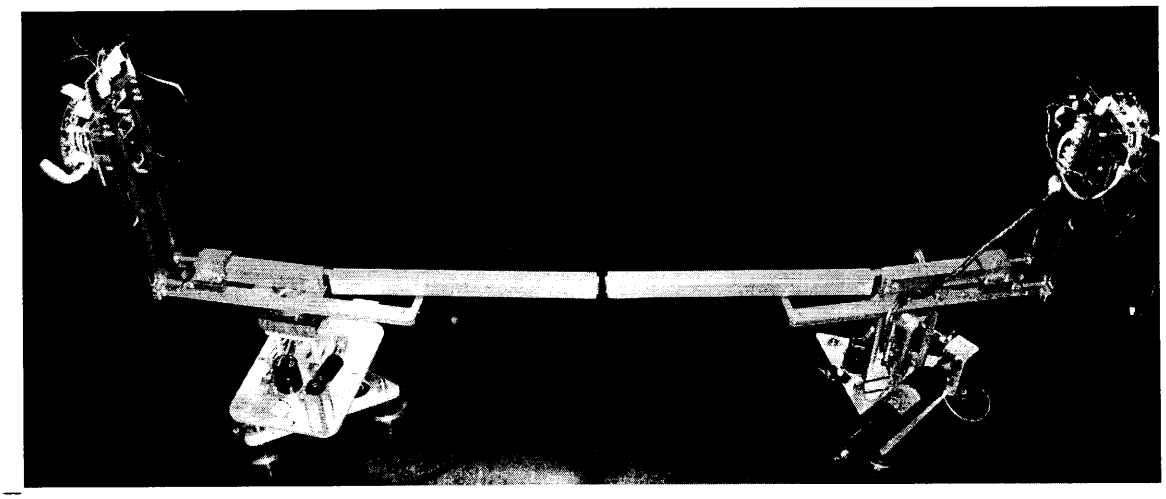


Figure 1.6: The SWARM flexible beam mounted to two air carriages.

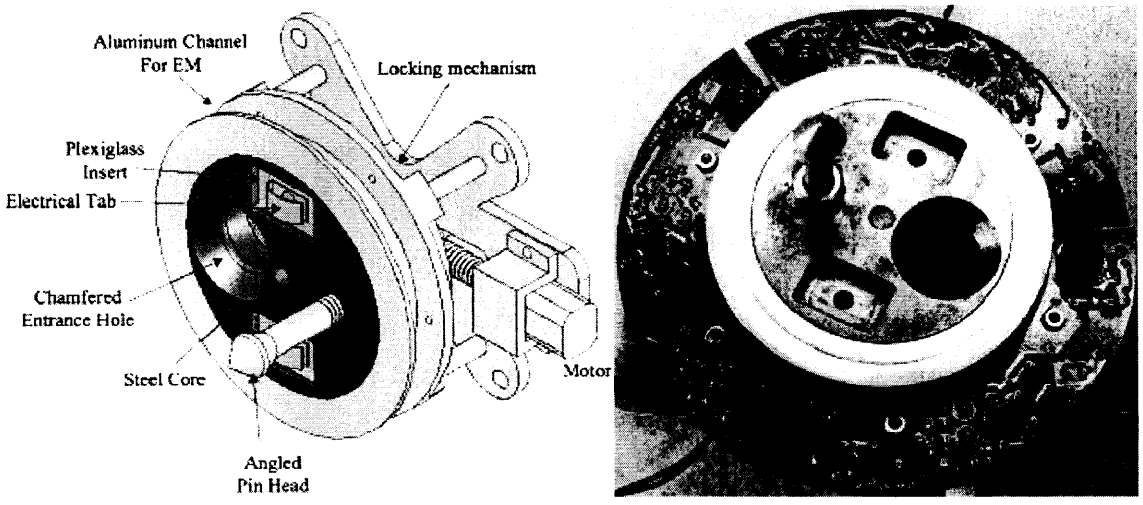


Figure 1.7: The SWARM Universal Docking Port with highlighted features. The UDP on the right has a ring of ultrasound beacons and receivers for relative metrology.

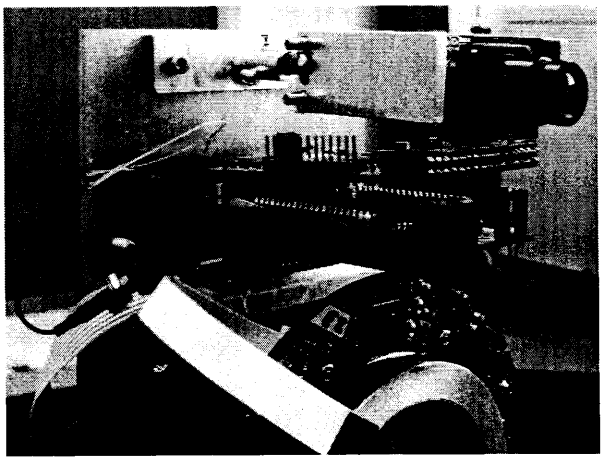


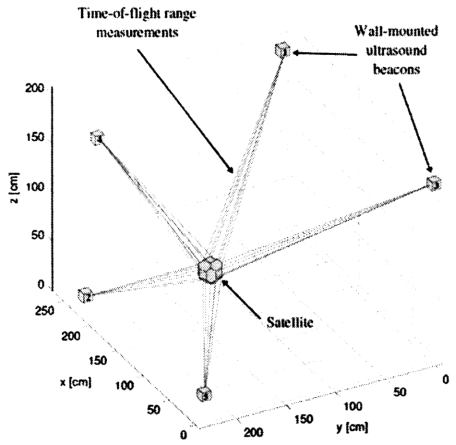
Figure 1.8: A Surveyor Corp. SRV-1 digital camera mounted above the UDP is used to observe the motion of the flexible beam.

may be provided by the GPS satellite constellation or differential position estimates calculated using sensors in the assembly workspace. The SWARM propulsion module uses the SPHERES pseudo-GPS global metrology system [18] for estimating its global position and orientation on the 2D flat floor.

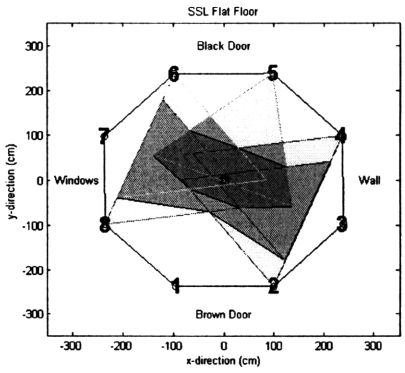
The metrology system consists of 5 ultrasound beacons placed at the perimeter of the work area in known reference positions. During a measurement cycle, the satellite combines time of flight information from each of the beacons in an Extended Kalman Filter (EKF) to produce a global position and orientation estimate.

During typical operations aboard the International Space Station, the global metrology beacons are arranged to form two intersecting planes as shown in 1.9a. This configuration aids the estimator in determining a consistent 6DOF state estimate. For the SWARM experiment, three beacons are sufficient to initialize the 2D position estimate, and two beacons are sufficient to maintain it. The beacons are placed to maximize the coverage area formed by three overlapping beacons as shown in 1.9b. Each beacon projects a swath 3m long and 60^o wide onto the floor corresponding to the maximum range and bearing for ultrasound measurements.

No additional modifications to the SPHERES global estimator were made to account for operation on the 2D flat floor. Several careful tests were performed to verify consistent convergence within the 2- and 3-beacon overlap zones. Typical position



(a) 5 beacons on the ISS are arranged in three dimensions for 6DOF measurements [18].



(b) The 5-beacon arrangement for the SWARM flat floor maximizes the 2- and 3-beacon coverage area.

Figure 1.9: Global metrology configuration for ISS and SWARM using 5 beacons.

errors of 0.5-1.5cm were on the order of those expected for the global metrology system. Had it been necessary to constrain the state estimate to the 2D surface of the flat floor to improve accuracy, two options could have been pursued: 1) develop a new estimator that only calculates the position and orientation state (x_b, y_b, θ_b), or 2) implement state constraints in the global estimator. For the second option, Simon and Chia suggest in [24] that an acceptable solution is to simply reset the known elements of the state vector, such as the height above the floor, to their know values at every estimation cycle.

1.4 Literature Review

The control of flexible structures has been explored from many different perspectives which might be classified in one of three major areas: flexible structure control, industrial robotics, and space-based robotic assembly.

Flexible structure control has the objective of eliminating structural vibrations to prevent damage or improve the performance of scientific instruments. Flexible space structures such as the International Space Station or communications satellites rely on flexibility control to stabilize their motion and suppress vibrations. For maneu-

vering, flexible structure control is primarily concerned with, at most, reorientation or following attitude trajectory profiles but is not focused on relative positioning in a workspace. For assembly of flexible structures, established techniques for vibration suppression, such as sliding mode control [25] and input shaping [15] contribute useful ideas.

Industrial assembly plants for automobiles and consumer products make extensive use of robotic manipulators. A wide body of literature has been developed for improving the performance of robotic manipulators in the presence of load uncertainty and manipulator flexibility. On-orbit assembly shares many similarities with the tasks performed by industrial robots, including transportation and precise positioning of parts. Some assembly scenarios even envision the use of lightweight versions of industrial robots. Given these similarities, it is useful to draw upon the techniques developed for industrial robots when implementing control and estimation algorithms for assembly spacecraft.

A particularly useful technology in industrial robotics is adaptive control. Given the kinematic layout of a manipulator (link lengths and relative orientations), adaptive controllers are capable of asymptotically tracking desired trajectories as if the dynamic parameters (masses and inertias) are known. With an established history in the world of rigid robot manipulators, extensions have been proposed for the control of robots with flexible joints [8] and even robots with uncertain kinematic properties [2]. Adaptive control for flexible systems is still a developing field because the problem falls in the realm of underactuated control, where the number of degrees of freedom exceeds the number of actuators. Many control techniques for underactuated control, such as partial feedback linearization [27] require a model inversion to calculate the required control commands. Inverting the model disrupts the important linear-in-the-parameters property that is exploited in many adaptive techniques, and therefore most methods rely on function approximation techniques like neural networks [21, 28] or basis functions [10] to perform the adaptation. The approach used in this study avoids the model inversion using a normal form augmentation approach proposed by Gu and Xu in [9] but includes additional insight from the feedback linearization and

function approximation methods.

The third area is the topic under consideration, the assembly of flexible structures. This area has elements of both of the two preceding groups. As a rigid body attached to a flexible appendage, the assembly robot has the same topology as a generic flexible satellite, but like a robotic manipulator, the robot must also maneuver its payload along a desired trajectory while compensating for vibrational disturbances. Some useful analogies exist, such as the Space Shuttle Remote Manipulator System, as well as a number of studies on free-flying and free-floating robot manipulator systems. Dubowsky et al. have conducted several laboratory studies, which include control [14], estimation [16], and cooperative assembly [6] of flexible structures using multiple free-flying robots. This study will use a single robot to perform the tasks of vibration estimation and flexibility control.

1.5 Summary of Approach

This thesis is divided into five chapters covering the development of a control and estimation system for the manipulation of a 2D flexible beam element. Chapter 2 constructs a dynamic model for an assembly robot attached to a flexible beam structure, along with simplified representations of the dynamics used in the remaining chapters. Chapter 3 introduces a vision-based estimation system for estimating the vibrational dynamics of the flexible beam as well as a simplified estimator for direct measurement of beam deflection. Chapter 4 presents three control approaches for maneuvering the beam along predefined trajectories. Chapter 5 concludes with observations and recommendations for future work.

Chapter 2

Dynamic Models

2.1 Overview

Before developing a control and estimation system for an assembly satellite, it is important to understand the dynamics of a translating, rotating base connected to a flexible structure. This chapter contains three main sections. The first derives a dynamic model for the SWARM segmented flexible beam and relates it to other flexible systems. The second section introduces a linearized model and a simplified model based on the full dynamic model, and the last section examines the models in simulation and hardware testing.

2.2 Derivation of Beam Dynamics

2.2.1 Review of the Euler-Lagrange Method

This section reviews the Euler-Lagrange method used in a wide variety of dynamics and control problems. The structure of the resulting equations have many useful features that are used throughout this study.

Given the Lagrangian,

$$L = T - V \tag{2.1}$$

with the kinetic energy T and potential energy V expressed in generalized coordinates

$\mathbf{q} \in \mathbb{R}^n$, the equations of motion can be developed from

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} = \boldsymbol{\tau}. \tag{2.2}$$

The generalized force $\tau \in \mathbb{R}^n$ is the input to the system. Dynamics derived from (2.2) are frequently arranged into the form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}. \tag{2.3}$$

The matrix $\mathbf{M} \in \mathbb{R}^{nxn}$ is a generalized inertia matrix that represents the masses and inertias of the links along with their dynamic couplings, $\mathbf{C} \in \mathbb{R}^{nxn}$ represents velocity-dependent, or Coriolis, terms, and $\mathbf{G} \in \mathbb{R}^n$ contains position-dependent torques such as from gravity or bending. For $\mathbf{C}$ defined as

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{n} \frac{\partial M_{ij}}{\partial q_k}\dot{q} + \frac{1}{2}\sum_{k=1}^{n} \left(\frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i} \right) \dot{q}_k \tag{2.4}$$

this representation has a *skew-symmetry* property [26] such that $\dot{\mathbf{M}} - 2\mathbf{C}$ is skew-symmetric. Therefore, for all $\dot{\mathbf{q}}$

$$\dot{\mathbf{q}}^T \left(\dot{\mathbf{M}} - 2\mathbf{C} \right) \dot{\mathbf{q}} = 0. \tag{2.5}$$

This well known property of (2.3) is the result of energy conservation and is frequently used in the development of nonlinear control laws for physical systems.

When developing the symbolic representation of (2.3) it is convenient to use the following relationships

$$\mathbf{M} = \frac{\partial}{\partial \dot{\mathbf{q}}} \left(\frac{\partial T}{\partial \dot{\mathbf{q}}} \right) \tag{2.6}$$

$$\mathbf{G} = \frac{\partial V}{\partial \mathbf{q}}. \tag{2.7}$$

Nonconservative forces can also be inserted as a fourth matrix $\mathbf{D}$. For example viscous

damping might be added as

$$\mathbf{D}(\dot{\mathbf{q}}) = diag(d_1, d_2, \ldots, d_n)\dot{\mathbf{q}} \tag{2.8}$$

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}. \tag{2.9}$$

.

2.2.2 Dynamics of the Flexible Segmented Beam

In this section the 2D dynamics of a free-floating planar robot connected to a flexible segmented beam are developed using the Euler-Lagrange method. Figure 2.1 displays the physical states involved in the dynamics of the 4-link SWARM robot. Defining $\mathbf{x}_b = \begin{pmatrix} x_b & y_b & \theta_b \end{pmatrix}^T$ as the global position and orientation of the robot base, and $\boldsymbol{\delta} = \begin{pmatrix} \delta_1 & \delta_2 & \delta_3 \end{pmatrix}^T$ as the joint deflections relative to the previous link, a candidate generalized coordinate system is

$$\mathbf{q} = \begin{pmatrix} \mathbf{x}_b \\ \boldsymbol{\delta} \end{pmatrix} = \begin{pmatrix} x_b \\ y_b \\ \theta_b \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{pmatrix} \tag{2.10}$$

This separation of coordinates has significance in the analogy between the segmented beam model and the Euler-Bernoulli model discussed in Section 2.2.4. The Lagrangian can be constructed by viewing the system as a series of connected rigid bodies, each with a center of mass m_i located at $\mathbf{r}_i$. Starting with the base position $\mathbf{r}_0$ and assuming the links are uniform and axially symmetric, the position of each

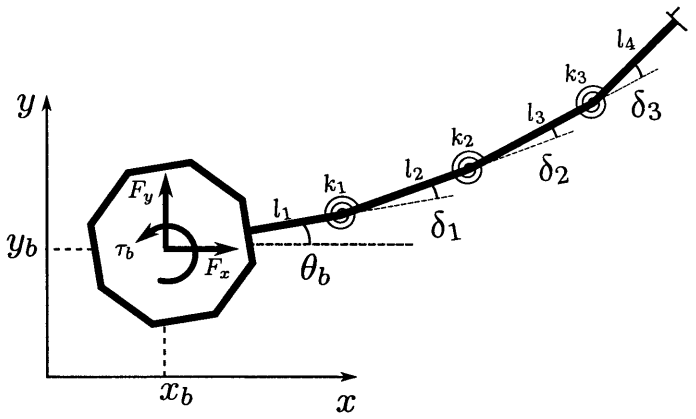


Figure 2.1: The important states of the SWARM system are the global position and orientation of the robot base and the deflection angles of each of the joints.

center of mass in the global frame is

$$\mathbf{r}_0 = \begin{pmatrix} x_b \\ y_b \end{pmatrix} \tag{2.11}$$

$$\mathbf{r}_i = \mathbf{r}_0 + \begin{pmatrix} \frac{1}{2} l_i \cos(\phi_i) + \sum_{k=1}^{i-1} l_k \cos(\phi_k) \\ \frac{1}{2} l_i \sin(\phi_i) + \sum_{k=1}^{i-1} l_k \sin(\phi_k) \end{pmatrix} \tag{2.12}$$

$$\phi_k = \theta_0 + \sum_{j=1}^{k-1} \delta_j \tag{2.13}$$

with l_k representing the length of link k. Differentiating these expressions with respect to time yields the translational and angular velocity of each body

$$\dot{\mathbf{r}}_0 = \begin{pmatrix} \dot{x}_b \\ \dot{y}_b \end{pmatrix} \tag{2.14}$$

$$\dot{\mathbf{r}}_i = \dot{\mathbf{r}}_0 + \begin{pmatrix} -\frac{1}{2} l_i \sin(\phi_i)\dot{\phi}_i - \sum_{k=1}^{i-1} l_k \sin(\phi_k)\dot{\phi}_k \\ \frac{1}{2} l_i \cos(\phi_i)\dot{\phi}_i + \sum_{k=1}^{i-1} l_k \cos(\phi_k)\dot{\phi}_k \end{pmatrix} \tag{2.15}$$

$$\omega_i = \dot{\phi}_i \tag{2.16}$$

Combining the terms for rotational and linear velocity, the kinetic energy is

$$T = \sum_{i=0}^{n} m_i \dot{\mathbf{r}}_i^T \dot{\mathbf{r}}_i + I_i \omega_i^2 \tag{2.17}$$

where, m_i is the mass of body i, and I_i is the moment of inertia of body i about its center of mass.

In 2D, the only potential energy in the system is stored in the bending of the torsional spring, and there is no gravitational potential energy. As a function of the link deflection, δ_i, and the spring constant k_i, the total potential energy is

$$V = \sum_{i=1}^{3} k_i \delta_i^2 \tag{2.18}$$

With both terms of the Lagrangian, Equations (2.6), (2.4), and (2.7) can be applied to extract the symbolic representations of **M**, **C**, and **G**. A MATLAB® script for performing this calculation is included in Appendix B.

2.2.3 Underactuated Dynamics

Having derived the left hand side of (2.3) in the previous section, we turn our attention to the right hand side. The thrusters for the SWARM robot can apply a force in the x and y directions as well as a torque about the base of the robot. Since there are no actuators in the flexible beam, the generalized force vector has the form

$$\boldsymbol{\tau} = \begin{pmatrix} \boldsymbol{\tau}_x \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} F_x \\ F_y \\ \tau_b \\ \mathbf{0} \end{pmatrix}.$$

Systems of this form are referred to as *underactuated* because there are fewer actuators than degrees of freedom. With this in mind, (2.3) can be partitioned into submatrices

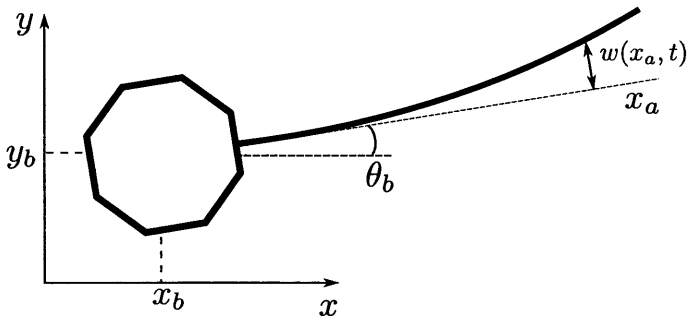


Figure 2.2: Translating and rotating base connected to an Euler-Bernoulli beam

corresponding to the actuated coordinates, $\mathbf{x}$, and the unactuated coordinates, $\boldsymbol{\delta}$.

$$\begin{bmatrix} \mathbf{M}_{xx} & \mathbf{M}_{x\delta} \\ \mathbf{M}_{\delta x} & \mathbf{M}_{\delta\delta} \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{x}}_0 \\ \ddot{\boldsymbol{\delta}} \end{pmatrix} + \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{x\delta} \\ \mathbf{C}_{\delta x} & \mathbf{C}_{\delta\delta} \end{bmatrix} \begin{pmatrix} \dot{\mathbf{x}}_0 \\ \dot{\boldsymbol{\delta}} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{G}(\boldsymbol{\delta}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{\tau}_x \\ \mathbf{0} \end{pmatrix} \quad (2.19)$$

The spring torques in $\mathbf{G}(\boldsymbol{\delta})$ are proportional to joint deflections $\boldsymbol{\delta}$ and can be factored as $\mathbf{G}(\boldsymbol{\delta}) = \mathbf{K}_f\boldsymbol{\delta}$ where $\mathbf{K}_f = diag(k_1, k_2, k_3)$ is a matrix of spring constants. Substituting into (2.19) results in

$$\begin{bmatrix} \mathbf{M}_{xx} & \mathbf{M}_{x\delta} \\ \mathbf{M}_{\delta x} & \mathbf{M}_{\delta\delta} \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{x}}_0 \\ \ddot{\boldsymbol{\delta}} \end{pmatrix} + \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{x\delta} \\ \mathbf{C}_{\delta x} & \mathbf{C}_{\delta\delta} \end{bmatrix} \begin{pmatrix} \dot{\mathbf{x}}_0 \\ \dot{\boldsymbol{\delta}} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_f \end{pmatrix} \mathbf{q} = \begin{pmatrix} \boldsymbol{\tau}_x \\ \mathbf{0} \end{pmatrix} \quad (2.20)$$

As shown in the next section, this structure is common to many systems with flexibility.

2.2.4 Relation to Euler-Bernoulli Beam Dynamics

Though the segmented beam only approximates the dynamics of a structure with distributed flexibility, the dynamic model in Equation (2.20) is nearly identical to the assumed modes method sometimes used for the analysis of flexible structures. To see the relationship between the two models, we consider a thin flexible beam of length L of uniform density per unit length ρ and bending stiffness EI connected to a moving base as shown in Figure 2.2. If the time-varying deflection from the body x-axis of the robot base is expressed as $w(x_a, t)$, the position of an arbitrary point on the beam

in the global frame is

$$\mathbf{r}(x_a) = \begin{pmatrix} x_b \\ y_b \end{pmatrix} + \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{pmatrix} x_a \\ w(x_a,t) \end{pmatrix} \tag{2.21}$$

To form the Lagrangian, the discrete sum over velocities of rigid links in Equation (2.17) becomes a continuous integral over infinitesimal mass elements

$$T = \frac{1}{2}\rho \int_0^L \dot{\mathbf{r}}^T(x_a)\dot{\mathbf{r}}(x_a)dx_a \tag{2.22}$$

and Equation (2.18) becomes an integral representing the elastic energy stored in bending

$$V = \frac{1}{2}\int_0^L EI\left(\frac{\partial^2 w}{\partial x_a^2}\right)^2 dx_a \tag{2.23}$$

Before developing the equations of motion, $w(x_a,t)$ is separated into temporal and spatial components

$$w(x_a,t) = \sum_{i=1}^{n} \phi_i(x_a)\delta_i(t) \tag{2.24}$$

where $\phi_i(x_b)$ are assumed mode shapes and $\delta_i(t)$ are their time-varying amplitudes. For a true flexible structure $n = \infty$, but the model is usually truncated after the first few modes for control design. The mode shapes ϕ_i in this system are chosen based on clamped-free boundary conditions because the beam is rigidly clamped to the robot base. The resulting equations of motion can be arranged in a form similar to Equation (2.20), with the elements of $\mathbf{M}$ and $\mathbf{C}$ determined from integral expressions of the mode shapes and their amplitudes. A thorough derivation for the dynamics of translating and rotating flexible beam using this method can be found in [31].

Given the shared structure between the segmented beam dynamics and the Euler-Bernoulli beam dynamics, the control and estimation algorithms presented in this study are applicable to both systems. We can also apply techniques from the large body of flexible dynamics research to the process of simplifying the dynamic model, as presented in the next section.

2.3 Simplified Dynamic Models

Though the modeling process above attempts to capture a detailed description of the flexible dynamics, certain simplifications can be introduced to aid the development of control and estimation algorithms. This section will examine two important model simplifications that will be used for further study in the next chapters. The first model is a linearized version of the true dynamic model to be used in a full state estimator, and the second is a simplified lumping approximation for further use in both estimation and control.

2.3.1 Linearized Dynamics

After developing the nonlinear dynamics in (2.20), the model can be linearized by substituting the equilibrium point $(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}$ into the matrices $\mathbf{M}$ and $\mathbf{C}$. Since $\mathbf{C} = \mathbf{0}_{nxn}$ at the equilibrium and $\mathbf{M}$ is always invertible, the linear dynamics in state-space form are

$$\begin{aligned} \mathbf{x} &= \begin{pmatrix} \mathbf{x}_b \\ \boldsymbol{\delta} \end{pmatrix} \\ \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \end{aligned}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{M}^{-1} \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \end{bmatrix} \tag{2.25}$$

The structure of the matrix $\mathbf{A}$ can be further refined by substituting $\mathbf{K} = diag(0, 0, 0, k_1, k_2, k_3)$ which eliminates the first three columns of $\mathbf{M}^{-1}$.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{6x3} & \mathbf{0}_{6x3} & \mathbf{I}_{6x6} \\ \mathbf{0}_{6x3} & \mathbf{H} & \mathbf{0}_{6x6} \end{bmatrix} \tag{2.26}$$

with $\mathbf{H}$ as the nonzero columns of $-\mathbf{M}^{-1}\mathbf{K}$. This form is particularly important for the observability characteristics of the full state estimator discussed in Section 3.3. The lower right submatrix may also be modified if the system includes any velocity-

dependent damping terms. Adding viscous damping as in Equation (2.8) adds the term $-\mathbf{M}^{-1}\mathbf{D}$ to the lower-right submatrix.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{6x3} & \mathbf{0}_{6x3} & \mathbf{I}_{6x6} \\ \mathbf{0}_{6x3} & \mathbf{H} & -\mathbf{M}^{-1}\mathbf{D} \end{bmatrix} \tag{2.27}$$

Since we can only consider the linear model to reflect the behavior of the dynamics for small perturbations about the linearization point, we lose several important relationships between the rigid body dynamics and the beam dynamics. The dynamics of the x_b coordinate are decoupled from all dynamics except the F_x actuator command, and applying a body torque does not result in a translation of the base as we would expect. Applying a force, F_y, causes a rotation, but the x_b coordinate is unaffected. Unless we choose an alternate coordinate frame for developing the linear model, the linearized dynamics should not be used to simulate global motion with large expected rotations.

As shown in Section 2.4.1, the vibrational dynamics are better represented in the linear model because the beam deflections tend to stay within the range of the small angle assumptions, and in the case of no damping, they are not affected by the translational dynamics. Therefore, we conclude that the linear model is primarily useful as a tool for modeling the flexible dynamics.

2.3.2 Lumped Flexibility Model

In this section, we consider a dynamic model of the flexible beam system with all flexible dynamics lumped into a single flexible joint as shown in Figure 2.3. This simplification is sometimes made in the analysis of satellites with flexible appendages or for a simplified approach to modeling flexible structures. For the purposes of docking the free end of the flexible beam we are primarily concerned with eliminating the deflection of the tip of the beam, so this approximation is a compact way of capturing the important behavior of the system. There is an implicit assumption that the behavior is well represented by a single vibrational mode though controllers

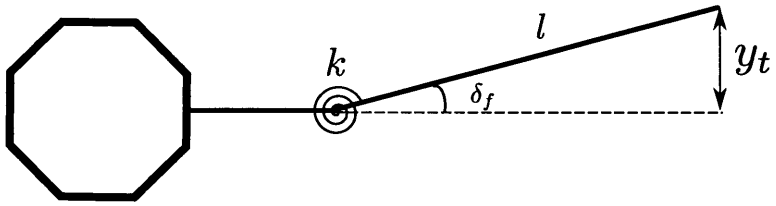


Figure 2.3: In the simplified flexibility model, all flexible dynamics are lumped into a single torsional spring attached to the rigid body of the satellite.

and estimators may still extract information about the other modes from the tip deflection because it is a weighted sum of the mode amplitudes.

A second motivation for developing a reduced model is parameter uncertainty. The flexible dynamics presented in the previous sections have assumed the availability of certain physical parameters such as link lengths, moments of inertia, and stiffnesses. These parameters may vary significantly across the variety of structural elements used for the assembly of a space structure, and even identical components may have different dynamic behavior if they are manipulated from different directions. This parameter uncertainty motivates the exploration of adaptive control techniques in Section 4.4 and the development of a single model with a small set of parameters that represents the important flexible dynamics for multiple systems.

Developing the dynamics of the simplified model, requires a guess for the spring constant, k, that approximates the behavior of the true model. The simplification presented here is similar to the method shown in [7] in which the spring constant is determined by matching the potential energy stored in a static deflection of the true flexible beam. To apply this method to the SWARM beam, we start with an expression for the tip deflection, y_t, as a function of the joint deflection angles $\boldsymbol{\delta} =$

$\begin{pmatrix} \delta_1 & \delta_2 & \delta_3 \end{pmatrix}^T$ and the link lengths

$$y_t = l_2 \sin(\delta_1) + l_2 \sin(\delta_1 + \delta_2) + l_4 \sin(\delta_1 + \delta_2 + \delta_3) \tag{2.28}$$

For small angles, the deflection is linear in $\boldsymbol{\delta}$

$$\mathbf{L} = \begin{bmatrix} (l_2 + l_3 + l_4) & (l_3 + l_4) & l_4 \end{bmatrix} \tag{2.29}$$

$$y_t = \mathbf{L}\boldsymbol{\delta} \tag{2.30}$$

and the simplified joint angle deflection, δ_f, is approximately

$$\delta_f = \frac{y_t}{l} \tag{2.31}$$

The potential energy in the true flexible beam can now be related to the potential energy stored in the torsional spring by

$$\frac{1}{2}\boldsymbol{\delta}^T \mathbf{K}_f \boldsymbol{\delta} = \frac{1}{2} k \left(\frac{y_t}{l}\right)^2 \tag{2.32}$$

This equation is underdetermined, but it can be solved by assuming the beam is temporarily in a static condition and therefore $\boldsymbol{\delta}$ minimizes the stored potential energy. After imposing the constraint from (2.30) so that the deflection matches the intended tip position, $\boldsymbol{\delta}$ is calculated by solving the following optimization

$$\min_{\delta} \frac{1}{2} \left[\boldsymbol{\delta}^T \mathbf{K}_f \boldsymbol{\delta} + \lambda \left(\mathbf{L}\boldsymbol{\delta} - y_t\right)\right] \tag{2.33}$$

which has the solution

$$\lambda = \frac{-y_t}{\mathbf{L}\mathbf{K}_f^{-1}\mathbf{L}^T}$$

$$\boldsymbol{\delta} = \frac{\mathbf{K}_f^{-1}}{\mathbf{L}\mathbf{K}_f^{-1}\mathbf{L}^T} \left(\mathbf{L}^T y_t\right) \tag{2.34}$$

Substituting into the left hand side of (2.32)

$$\begin{aligned}\frac{1}{2}\boldsymbol{\delta}^T\mathbf{K}_f\boldsymbol{\delta} &= \frac{1}{2}\boldsymbol{\delta}^T\frac{(\mathbf{L}^T y_t)}{\mathbf{L}\mathbf{K}_f^{-1}\mathbf{L}^T} \\ &= \frac{1}{2}y_t^2\left(\frac{\mathbf{L}}{\mathbf{L}\mathbf{K}_f^{-1}\mathbf{L}^T}\right)\left(\frac{\mathbf{K}_f^{-1}\mathbf{L}^T}{\mathbf{L}\mathbf{K}_f^{-1}\mathbf{L}^\mathbf{T}}\right) \\ &= \frac{1}{2}y_t^2\left(\frac{1}{\mathbf{L}\mathbf{K}_f^{-1}\mathbf{L}^\mathbf{T}}\right)\end{aligned} \tag{2.35}$$

Comparing to the right hand side, the spring constant is

$$k = \frac{l^2}{\mathbf{L}\mathbf{K}_f^{-1}\mathbf{L}^T} \tag{2.36}$$

This completes the derivation of the potential energy term required for the calculation of the Lagrangian dynamics. It is important to note this approach is only an initial guess for a spring constant. Additional techniques such as system identification can be used to refine the estimate.

For the kinetic energy term, we will assume that the mass and inertia properties of the second link in the simplified model are the same as if the remaining links in the original model were a rigid rod. From here, the derivation of the dynamics follows the same procedure as Section 2.2.2. The resulting equations have a reduced generalized coordinate system

$$\mathbf{q} = \begin{pmatrix} x_b & y_b & \theta_b & \delta_f \end{pmatrix}^T \tag{2.37}$$

as well as a smaller partitioned dynamics

$$\begin{bmatrix} \mathbf{M}_{xx} & \mathbf{M}_{x\delta} \\ \mathbf{M}_{\delta x} & m_{\delta\delta} \end{bmatrix}\ddot{\mathbf{q}} + \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{x\delta} \\ \mathbf{C}_{\delta x} & C_{\delta\delta} \end{bmatrix}\dot{\mathbf{q}} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & k \end{bmatrix}\mathbf{q} = \begin{pmatrix} \boldsymbol{\tau}_x \\ 0 \end{pmatrix} \tag{2.38}$$

2.4 Model Verification

This section presents several comparisons in simulation between the models presented above and hardware tests to assess the models. All simulation tests were performed

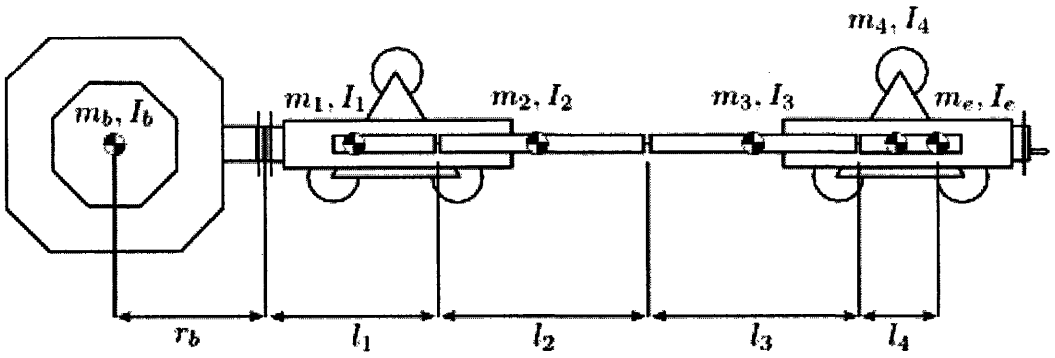


Figure 2.4: The simulation model is constructed from 6 bodies, shown here with their physical properties.

Property	Description	Value
$[\ l_1 \quad l_2 \quad l_3 \quad l_4\ ]$	length of links	$[\ 0.28 \quad 0.38 \quad 0.38 \quad 0.05\ ]\ m$
r_b	radius of prop. module	$0.27\,m$
$[\ m_1 \quad m_2 \quad m_3 \quad m_4\ ]$	link masses (m_1 includes UDP)	$[\ 6.64 \quad 0.18 \quad 0.18 \quad 0.03\ ]\ kg$
m_b	prop. module mass	$19.2\,kg$
m_e	UDP and end air carriage mass	$6.5\,kg$
$[\ I_1 \quad I_2 \quad I_3 \quad I_4\ ]$	rotational inertia of links	$[\ 0.1 \quad 0.002 \quad 0.002 \quad 5E{-}5\ ]\ kg \cdot m^2$
I_e	rot. inertia of UDP & carriage	$0.1\,kg \cdot m^2$
$[\ k_1 \quad k_2 \quad k_3\ ]$	torsional spring constants	$[\ 0.2 \quad 0.2 \quad 0.2 \quad 0.2\ ]\ \frac{N \cdot m}{rad}$

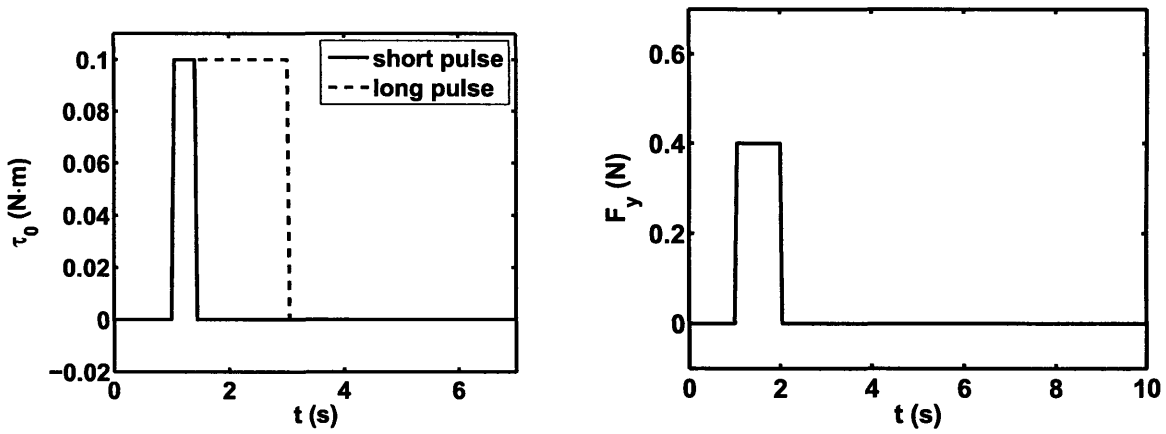
Table 2.4.1: Summary of the physical properties used for SWARM simulations

with the same physical properties, shown in Table 2.4.1, which approximately match the measured values for the SWARM hardware. A diagram of the hardware and the associated physical properties is shown in Figure 2.4. Each center of mass marker represents a separate body with a mass and moment of inertia.[1]

2.4.1 Simulation Testing

For simulation testing, a nonlinear multi-body baseline simulation was implemented in Simulink® SimMechanics™. As in the analytical dynamic model, the first link is rigidly connected to the propulsion module, and each of the successive links are connected by torsional springs. The last link is rigidly connected to a final body that represents the UDP and end air carriage. This model serves as the "true" dynamics and the remaining dynamic models are compared to this model for accuracy.

[1]Link l_4 is shorter than the other links because the docking port and air carriage at the end of the beam are considered a separate body.



(a) (b)

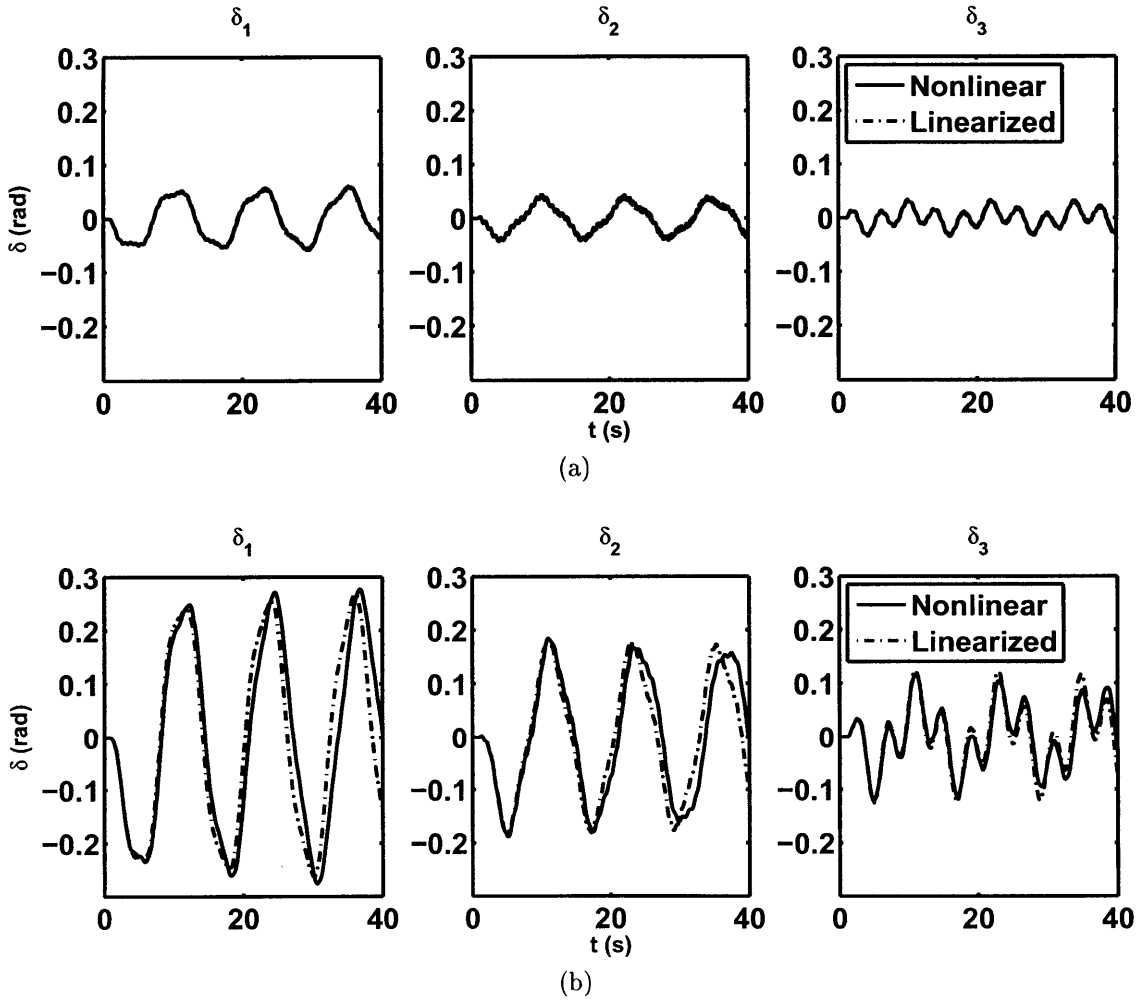
Figure 2.5: (a)Two torque profiles and (b) translation profile applied to the models for comparison. The short torque profile is representative of short bursts during a control cycle, and the longer profile is intended to show model differences.

The following sections compare the simulated responses for each system to the two torque inputs shown in Figure 2.5. The first input is a short torque pulse of 0.4 seconds that excites beam dynamics but keeps the response within the realm of small deflections, and the second profile is a longer pulse of 2 seconds that causes large deflections. Both torque pulses simulate the approximate strength of the SWARM thrusters at 0.1 N·m. The translation input simulates a linear thrust input with a 1 second pulse in the direction transverse to the beam. Though the true hardware system is heavily damped due to friction on the flat floor surface, these simulations are presented without damping to compare the vibration response in each of the dynamic models.

Joint Dynamics Comparison

These tests examine the ability of the linear model to approximate the joint dynamics of the nonlinear model. The simplified model is omitted because it only contains one flexible joint. Figure 2.6 compares the response of the linear and nonlinear models in response to the two torque profiles for each of the internal joint variables $\boldsymbol{\delta}$ =

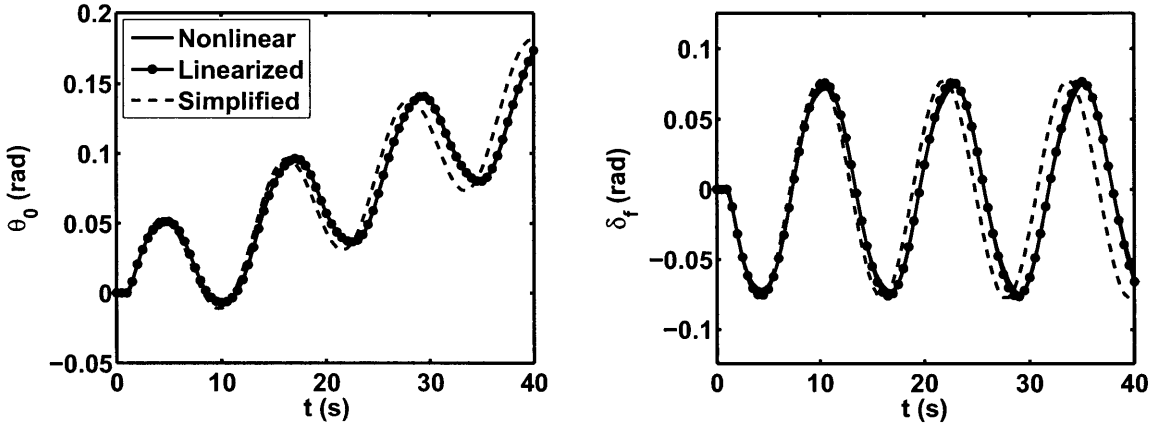
$\begin{pmatrix} \delta_1 & \delta_2 & \delta_3 \end{pmatrix}^T$. Under small deflections up to about 5 degrees for each joint, the linear model agrees very closely with the nonlinear model. With the longer torque profile, the linear model begins to show some discrepancies as deflection magnitudes exceed 10 degrees (0.17 rad). From these results we can conclude that the linear model will serve well for approximating the nonlinear flexible dynamics in the small angle regime.



(a)

(b)

Figure 2.6: (a) The linear model follows the nonlinear model well for deflections under 5 deg (0.09 rad). (b) For larger deflections above about 10 deg (0.17 rad) there are noticeable differences in amplitude and frequency.



(a) (b)

Figure 2.7: Simulated response of (a) rotation angle and (b) deflection angle to a 0.4 second torque input.

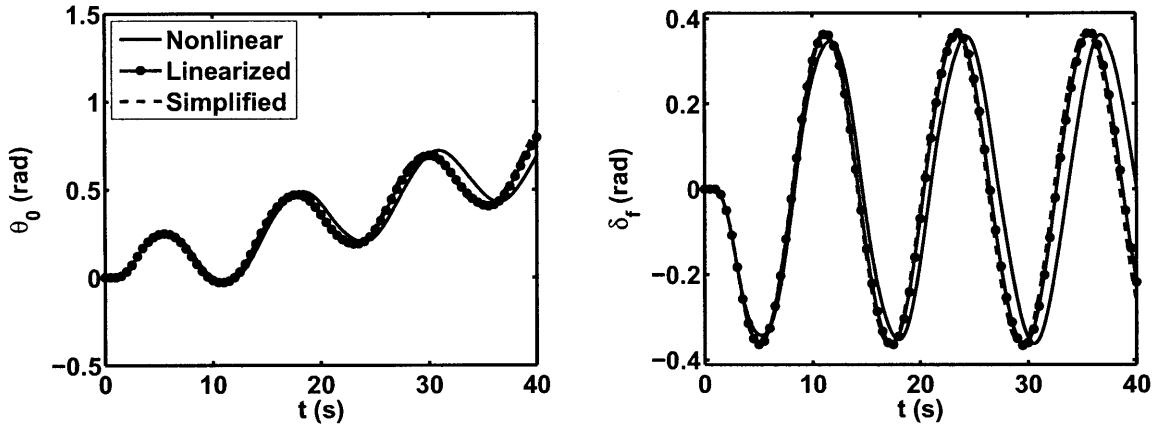
Rotation and Deflection Response

Figure 2.7 compares the responses of the simplified model, the linearized model, and the nonlinear model to the two torque signals. All models track the rigid body rotation induced by the torque input as shown by the increasing trend in Figure 2.7a, but the simplified model exhibits a slightly lower frequency vibration response. This is also evident in the comparison of tip deflection angles in Figure 2.7b, suggesting that the approximation for k in Equation 2.36 may need to be refined. For both measurements, the linear model is a very good approximation.

For the longer profile, we can examine the model accuracy for larger tip deflections. In Figure 2.8 the linear model begins to show some discrepancies as deflections exceed about 15 degrees (0.26 rad), while the simplified model continues to lag the true response. Again, we see that the linear model holds for smaller deflections.

Translation Response

Figure 2.9 examines the response of the models to a translation force input applied in the y direction, which is perpendicular to the beam direction. All models respond appropriately in the y direction, but the linearized model does not accurately reflect the movement in the x direction. This is expected because the global positions in



(a) (b)

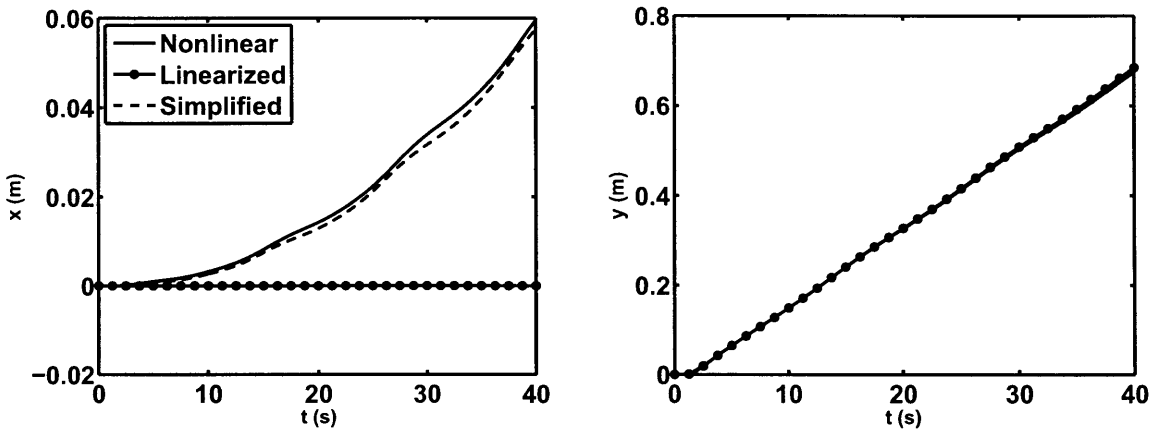
Figure 2.8: Simulated response of (a) rotation angle and (b) deflection angle to a 2 second torque input

the linearized model are decoupled from the rotational dynamics. If a more accurate global linear model were required, a better choice of reference frame might be the center of mass frame where the translational forces can be integrated independently from the vibrational dynamics. Since the vibrational dynamics remain reasonably accurate, the base position could be determined by applying a transformation from the center of mass frame to the base frame as a function of the joint angles and physical parameters.

Examining the response of the simplified model, we can see that the nonlinear coupling is maintained for translation. There are small errors in the exact position of the base, but the model continues to capture the primary dynamic behavior.

Simplified Model with System Identification

Simulation results in the previous sections suggest that the expression for the spring constant (2.36) is only a rough approximation of the best fit for the nonlinear dynamics. Using the input/output data generated from the simulations along with the symbolic expressions for the simplified beam dynamics it is possible to run a grey-box system identification for the uncertain parameters. In this case we are interested in identifying the spring constant k, and we add a fictitious end mass m_e to account for



(a) (b)

Figure 2.9: Translational response in the x and y directions

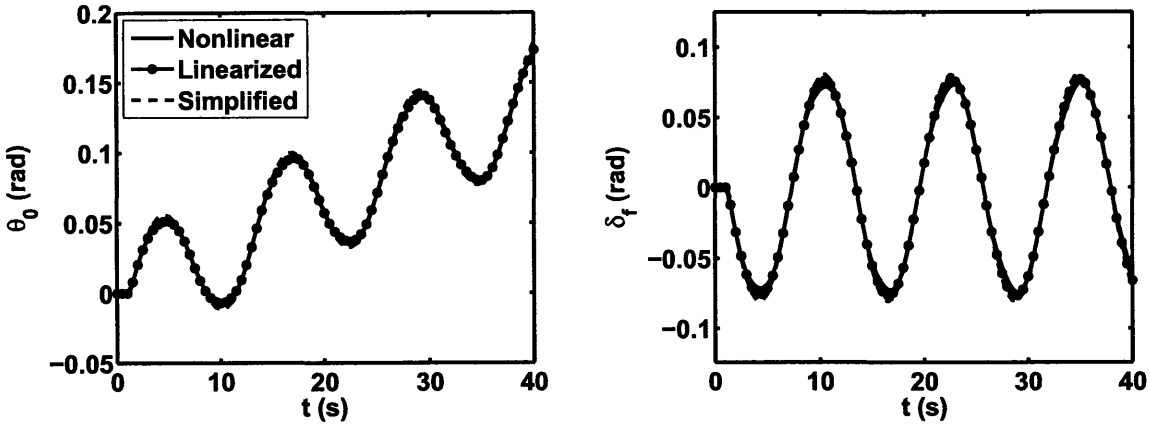
kinetic energy errors introduced by the single link approximation. Figure 2.10 shows the improvement in the response of the simplified model to the short torque profile. In particular, the frequency of oscillation more closely matches the full dynamic model.

While it is important that a set of parameters can be found to match a specific system response, we must also check that the parameters apply to other profiles and other system outputs. In Figure 2.11, the identified model is subjected to the long torque profile and the translation input. In these tests, the model continues to provide a good approximation. This suggests that the identified parameters provide the correct values for the approximate dynamic model.

We have now shown that the linear model provides good estimates of the internal dynamics of the flexible beam but does not capture the dynamic behavior under large motions. This makes the linear model suitable for a beam dynamics estimator but not for a trajectory controller. The simplified nonlinear model manages to capture the primary flexible behavior as well as the couplings with rigid motion. These attributes make the simplified model desirable as a template for a model-based control system.

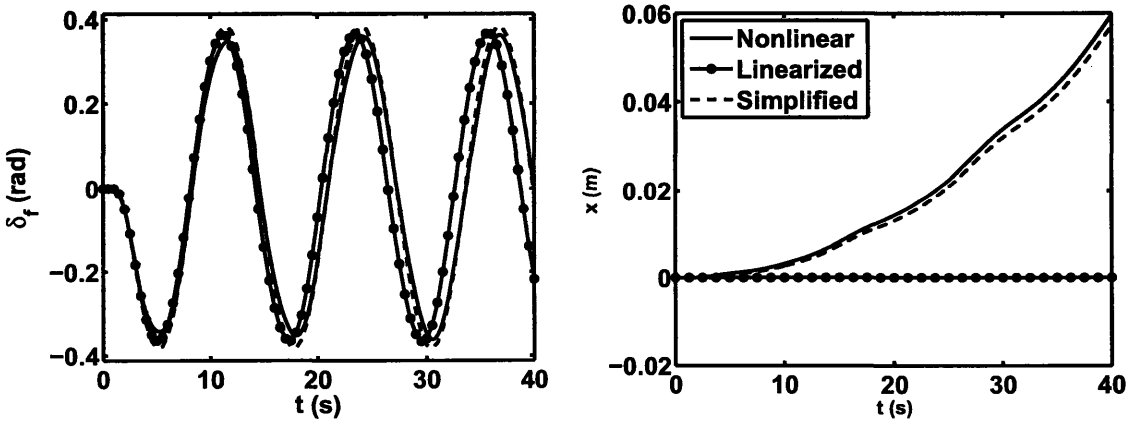
2.4.2 Hardware Testing

The purpose of the testing was to ensure that the simplified model could represent the flexible dynamics of a physical system in two different configurations. Under the



(a) (b)

Figure 2.10: The response of the simplified model to the short torque profile improves after system identification



(a) (b)

Figure 2.11: The parameters from system identification are also valid for the long torque profile (a) and the translation response (b)

limitation of using only beam deflection angle and SPHERES sensors as a measurement, the best model for comparison is the simplified beam model. To account for friction, an additional viscous damping term is added to the equations of motion as in Equation (2.8). This approach is consistent with previous friction modeling for SPHERES [4].

The testing environment for SWARM includes several challenges:

- **Friction** The air-bearing carriages used to support the propulsion module and beam are subject to friction due to drag, surface irregularities, and any small particles that may be present on the sliding surface. The main effect on the dynamics is the addition of significant damping, though occasional sticking can result in large step disturbances.

- **Parameter Uncertainty** Though most of the components in the system were measured for key parameters such as moment of inertia and mass, other parameters such as center of mass location and spring constants are uncertain.

- **Measurement Uncertainty** As shown in Chapter 3, with addition of unknown damping effects and parameter uncertainty, it is particularly difficult to develop an estimate for the joint deflections of the vibrating beam from the measurements available to SWARM. For hardware testing, the beam deflection angle is measured instead using the beam deflection estimator presented in Section 3.4.

- **Thruster Uncertainty** The thruster performance for SPHERES satellites has been thoroughly studied in [3], but is not as well known for the SWARM propulsion module due to a different physical layout and significantly higher mass flow rates. The estimates for thruster performance (in Table 1.3.1) were calculated from several short tests of angular acceleration.

Clamped Vibration Response

In this test, the free end of the beam was allowed to float on an air carriage while the end attached to the propulsion module was fixed in place. In this configuration, the

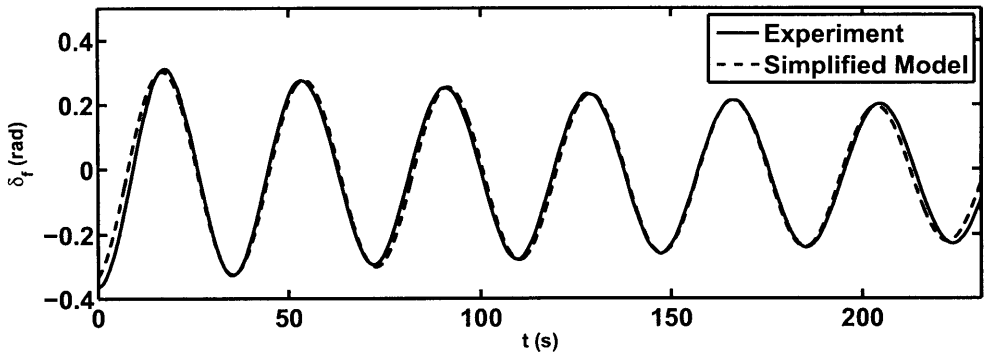


Figure 2.12: Simulation of the first second order system in Equation (2.39) vs. the actual responses for the clamped beam.

simplified beam dynamics reduce to a linear system

$$I\ddot{\delta_f} + b\dot{\delta_f} + k\delta_f = 0 \tag{2.39}$$

where I is the equivalent moment of inertia of the beam about the rotation point. Assuming that the moment of inertia is known, the test provides candidate values for the damping and spring constant terms after fitting the model to the data. Figure (2.12) compares the actual and simulated responses after fitting. The best-fit parameters for this test are $b = 0.02\,\frac{N \cdot m - s}{rad}$ and $k = 0.14\,\frac{N \cdot m}{rad}$ for an inertia value of $4.8\,kg \cdot m^2$.

Free-Free Vibration Response

For this test the propulsion module was allowed to float freely, and an oscillation was started by hand. To determine the parameters for this system, the spring constant, the damping constant, and the end mass were allowed to vary in the parameter identification. Figure 2.13 shows the response of the model for beam deflection and angular rate after fitting. The best-fit parameters are different in this test, as shown in Table 2.4.2, but the simulation and model are in close agreement. There are several possibilities for the discrepancies between the two test responses. Both data sets supply only a single frequency, so there may not be sufficient excitation to identify all parameters. Redundant parameters can also cause identification errors, as in the case

Parameter	Value
b_δ	$0.015 \frac{N \cdot m - s}{rad}$
b_θ	$0.002 \frac{N \cdot m - s}{rad}$
k	$0.18 \frac{N \cdot m}{rad}$
m_e	$6.58\,kg$

Table 2.4.2: Parameters for Free-Free Vibration Response

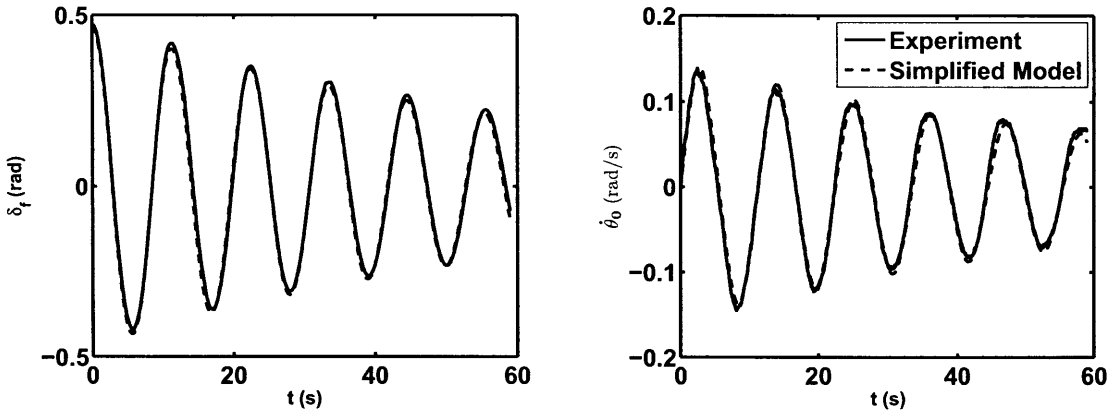


Figure 2.13: Simplified beam model simulation and measured values of (a) deflection angle and (b) base rotation rate for a free-free vibration response

above where uncertainty in the inertia approximation will corrupt the accuracy of the damping and spring constant estimates. Only the ratio between these quantities can be determined from the test. Ultimately, however, knowing the exact values of the parameters is not necessarily essential to providing accurate control. For adaptive control, as long as a parameter set can be adjusted to achieve the desired behavior, the exact values of the parameters are not needed. In this sense, the simplified model provides a compact form for representing a variety of systems.

2.5 Conclusions

Several models of varying complexity have been developed using the Euler-Lagrange method. The forms of these models have applicability beyond the segmented beam model of the SWARM hardware, and in particular with the Euler-Bernoulli model of

a uniform flexible beam. The segmented model itself lends some interesting insight into approximations for flexible dynamics, specifically the utility of a simplified model with a single flexible joint.

Simulation results show that a linear model is quite accurate in the small deflection angle regime. Given sufficient knowledge of the system parameters, we could use the model along with the wealth of tools in linear control analysis to build a control solution. However, as shown in the SWARM hardware testing, thoroughly identifying physical parameters requires more than a set of simple open loop tests and may not be feasible for a large number of distinct components. We are also fundamentally limited to performing initial analysis of the system on the ground where the laboratory environment introduces unwanted disturbances. While disturbances and uncertainties are detrimental to comparing models, they are analogs for the uncertainty faced while manipulating multiple components during an on-orbit assembly. As we saw in both the simulation and hardware testing, the simplified model has the important characteristic of being able to match a variety of flexible systems with the update of only a few parameters. This characteristic suggests the simplified model will be useful in an on-line adaptive control system that may be able to compensate for the uncertainty between components. Given the single measurement required to determine the beam deflection angle, the simplified model is also attractive for minimizing the sensor count and estimation complexity.

Chapter 3

Vision-Based Estimation for Flexible Space Structures

3.1 Overview

While manipulating a flexible element, an assembly vehicle requires knowledge of its position within the assembly workspace as well as an estimate of the vibrational dynamics of the element, as shown in Figure 3.1. Collecting this information is challenging because the use of a propulsion tug for assembly poses restrictions on the manner in which flexible vibration measurements can be collected. Unless power and data connections are included in the docking interface or a wireless sensor network is constructed, it is not possible to embed active sensors in the structural element. This chapter presents methods for estimating flexible dynamics using vision-based and inertial sensors attached to the assembly robot. This method is capable of estimating the internal beam dynamics although it is sensitive to disturbances and to an uncertain physical model of the beam dynamics. To address robustness issues and reduce the estimator complexity, a simplified filter for estimating beam deflection angle is also presented.

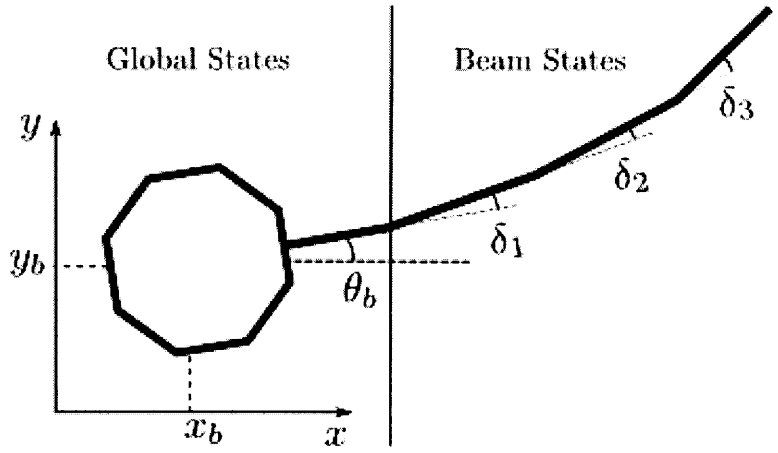


Figure 3.1: Global states determine the position and orientation of the robot in the workspace while the beam states determine the beam shape.

3.2 Beam Measurements

3.2.1 Vision-Based Beam Measurements

To allow for docking and undocking, all sensors involved in estimating the state of the flexible beam must be mounted to the propulsion tug. Cameras provide a cheap, lightweight method for passively extracting information about motion in a scene. For vibration estimation, the camera can provide position or angular measurements for uniquely identified features attached to the oscillating body. The SWARM testbed uses a camera to observe the motion of a small infrared LED at the tip of the flexible beam. A tip-mounted LED is useful for supplying information about vibrational motion as well as positioning for docking. Practical implementations could use fiducial markers or reflectors, which have been used in vision applications on-orbit, or even an LED powered by a small solar panel.

The first steps of the camera measurement process are shown in Figure 3.2. Beam measurements start as grayscale pixel intensity values referenced from the upper left corner of the image plane. Digital imaging CCDs are very sensitive to the infrared spectrum, and the light from the LED easily saturates the pixels near its position in the image frame. By adjusting the camera exposure and admitting only pixel intensities above a certain threshold, the LED can be made to appear as a dot in the

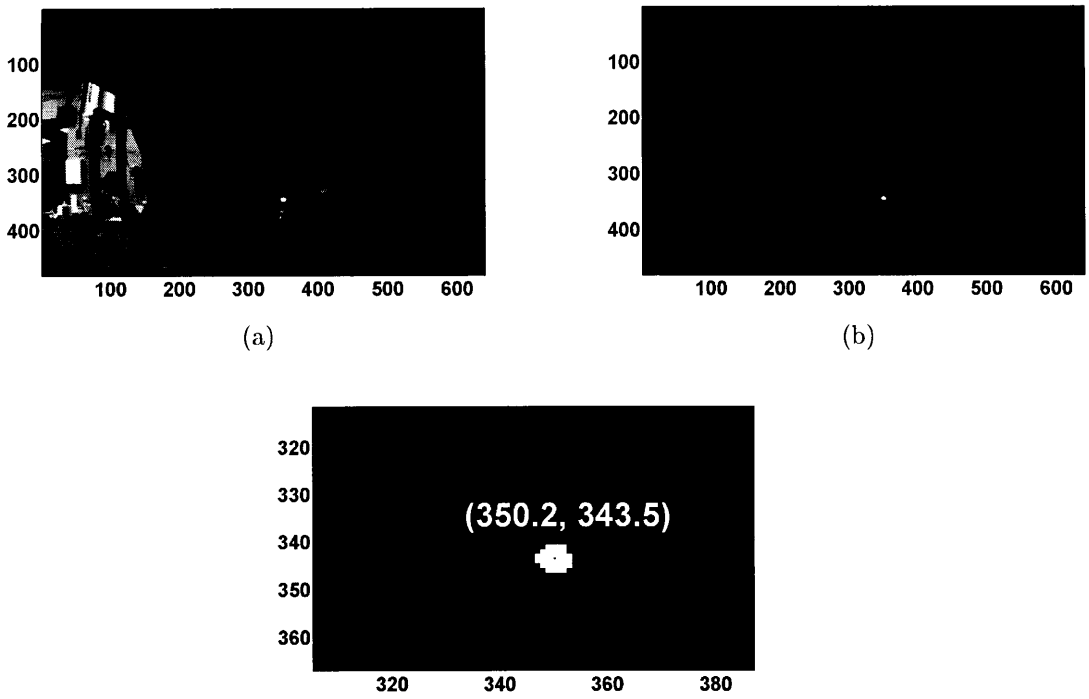


Figure 3.2: An example of the centroidng process: (a) An initial image is acquired with a low exposure, then (b) all pixels below a threshold are set to 0, and finally, (c) the centroid coordinates are determined from the remaining pixels (shown zoomed in on the LED)

image plane. The center of area of the dot can be found with accuracy less than or equal to a single pixel using a common centroiding algorithm such as

$$\begin{aligned} \bar{x} &= \frac{\sum_{i,j} i \cdot I_{i,j}}{\sum_{i,j} I_{i,j}} \\ \bar{y} &= \frac{\sum_{i,j} j \cdot I_{i,j}}{\sum_{i,j} I_{i,j}} \end{aligned} \tag{3.1}$$

The LED center position $(\bar{x}, \bar{y})$ is calculated by weighting each pixel coordinate with its intensity $I_{i,j}$ and dividing by the total intensity of the image. This is equivalent to calculating the first moment of intensity about the origin of the image plane.

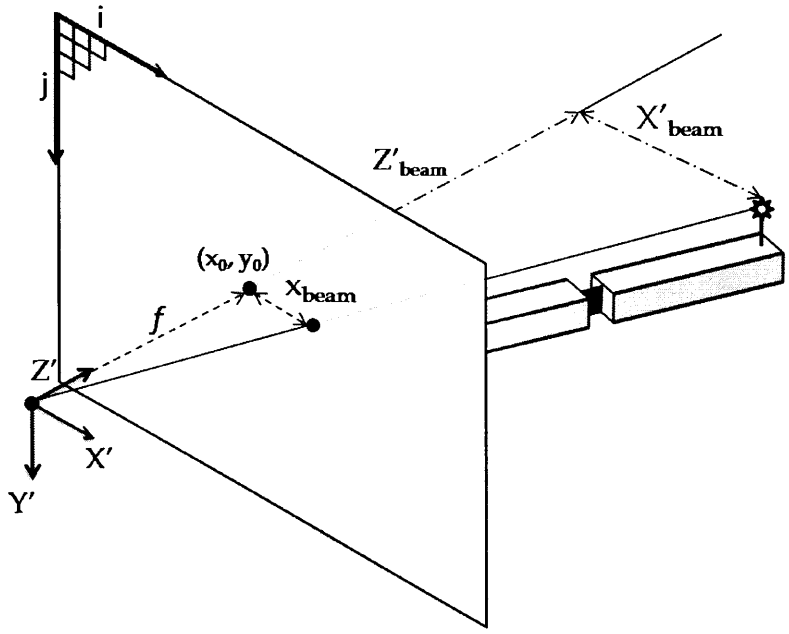


Figure 3.3: Camera measurements start as pixel intensities measured in the image frame and end as a beam deflection in the camera frame.

Next, the perspective projection (pinhole camera) equations [13] are applied to the image to convert from image frame pixel coordinates to camera frame distances.

$$\frac{X'}{Z'} = \beta\frac{\bar{x} - x_0}{f} \tag{3.2}$$

$$\frac{Y'}{Z'} = \beta\frac{\bar{y} - y_0}{f} \tag{3.3}$$

Subtracting the center of projection (x_0, y_0) shifts the centroid coordinates from the top-left reference frame to an image-center reference frame where they are scaled to a distance by the pixel size β. The remaining parts of equations (3.2) and (3.3) come from the similar triangles that share a common vertex at the focal point. Examining Figure 3.3, we can see that the ratio of the image plane coordinate to the focal length f is the same as the ratio of the camera frame coordinate to the depth Z'. At this point it is important to note that the perspective projection equations are 2 equations in 3 unknowns, so the camera frame distances (X', Y') are only known to within a scale factor. There are two ways of recovering the beam deflection X'_{beam}, both of which require a known measurement.

Y' **known** The first method is to use the known vertical displacement between the camera and the LED, which corresponds to the distance Y'. With Y' and $\bar{y}$, (3.3) can be solved for Z' to recover the scale factor in Equation (3.2).

Z' **known** The second approach makes the assumption that the depth of the LED in the camera frame Z' remains roughly constant and is equal to the length of the beam. In this case, the X' and Y' distances can be solved separately by substituting Z'_{beam} for Z'.

During SWARM testing, the first approach was initially attractive because the Y' distance should remain constant for 2D beam deflections. However, small rotations of the camera about the X' axis can cause large changes in the true Y' distance, and the method becomes particularly sensitive to small camera misalignment errors and vibrations of the camera mount. This leaves the second approach using the constant depth assumption. Equation (3.4) and Table (3.2.1) examine the error introduced by assuming a constant depth when the camera observes a deflection by θ degrees in the (X', Z') plane.

$$\frac{X'_{actual} - X'_{est}}{X'_{actual}} = (1 - \cos\theta) \tag{3.4}$$

For a 1 meter beam, deflections of up to 20 centimeters will cause a measurement error of less than a millimeter. With this calculation in mind it is reasonable to conclude that the approximation will not introduce significant error in the estimation of beam deflection. Other sources of error such as image noise are more likely to have a direct effect on the beam measurements. For instance, a 1 meter beam with a typical image noise of 0.8 pixel standard deviation will result in a deflection error with 1.5 millimeter standard deviation.

The last step of the measurement process is to transform from the camera frame to the body frame of the vehicle. Depending on the camera mounting arrangement both a rotation and a translation may be required. The relationship between a set of camera coordinates $\mathbf{x}' = \begin{bmatrix} X' & Y' & Z' \end{bmatrix}^T$ and body coordinates $\mathbf{x} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$

Deflection (deg)	Error (%)
5	0.4
10	1.5
15	3.4
20	6.0

Table 3.2.1: Error introduced by assuming a constant LED depth.

is expressed by the linear transformation:

$$\mathbf{x}' = \mathbf{R}\left(\mathbf{x} + \mathbf{r}_{oc}\right) \tag{3.5}$$

and the inverse transform:

$$\mathbf{x}' = \mathbf{R}^T\mathbf{x}' - \mathbf{p}. \tag{3.6}$$

$\mathbf{R}$ is the rotation matrix that rotates body frame coordinates into camera frame coordinates, and $\mathbf{r}_{oc}$ is the position of the camera in the body frame.

The SWARM system uses a simple configuration with the Z' axis along the body X axis of the propulsion module (aligned with the beam). In this arrangement, the beam deflection in the body frame, Y, is given by (3.8).

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \mathbf{x}' - \mathbf{r_{oc}} \tag{3.7}$$

$$Y = -X' \tag{3.8}$$

3.2.2 Other Sources of Measurement

Vision-based methods are not the only way to passively collect information about the beam dynamics. With one end rigidly clamped to the satellite, the beam is dynamically coupled to the satellite, and vibrations cause translations and rotations of the satellite base. Both motions can be measured by any inertial sensors that may be part of the robot's internal navigation system. Rotations of the base can be measured by gyros and translational accelerations by accelerometers. For this

study, only gyro measurements are considered because they are linearly involved in the state estimate, and they provide a high frequency measurement. Global position measurements such as the ultrasound metrology system in the SWARM project can also be used to observe both the rigid body motion and the effects of flexibility, but they tend to be low frequency and often involve nonlinear measurements. Other researchers have considered additional sensors, such as force-torque sensors at the interface between the assembly robot and the flexible structure [14].

3.3 Linear Kalman Filter

In addition to measuring the deflection of the flexible element, it is important for many control systems to provide a complete state estimate that includes the state variables of the flexible dynamics. For a generic flexible system, this amounts to estimating the time-varying amplitudes of the flexible modes, $\delta_i(t)$ (see Equation (2.24)), in the truncated dynamic model, and for the SWARM testbed, estimating the deflection angles, $\boldsymbol{\delta}$, of the flexible joints. As we saw in Chapter 2, in the small deflection angle regime, the linearized dynamic model is a good approximation for the nonlinear dynamics. This feature will be used to develop a traditional linear Kalman Filter that extracts the internal beam dynamics from tip motion and gyro measurements.

3.3.1 Measurement Equations

Section 3.2 detailed the transformation from camera measurements to beam deflection, but a linear measurement model must still be derived that relates the beam deflection to the estimated states. The objective is to determine the matrix $\mathbf{C}$ in the measurement equation[1]

$$\mathbf{y} = \mathbf{C}\mathbf{x}.$$

[1]Here $\mathbf{y}$ represents the vector of outputs, not to be confused with the vertical image position y or the body frame deflection Y.

For an Euler-Bernoulli beam, the linear measurement model uses the shape modes from Equation (2.24) evaluated at the end of the beam:

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_1(L) & \cdots & \phi_n(L) \end{bmatrix}^T \tag{3.9}$$

$$Y = \boldsymbol{\phi}^T \boldsymbol{\delta} \tag{3.10}$$

$$\mathbf{C} = \boldsymbol{\phi}^T \tag{3.11}$$

For the segmented beam deflection measurement equation, we start with the expression for the tip deflection from Section 2.3.2, assuming the LED is mounted at the end of l_4

$$Y = l_2 \sin(\delta_1) + l_2 \sin(\delta_1 + \delta_2) + l_4 \sin(\delta_1 + \delta_2 + \delta_3) \tag{3.12}$$

where l_i is the length of link i. After expanding the angle sum and assuming the angles δ_1, δ_2, and δ_3 are small,

$$Y \approx (l_2 + l_3 + l_4)\,\delta_1 + (l_3 + l_4)\,\delta_2 + l_4\delta_3. \tag{3.13}$$

Therefore, the linear measurement matrix for the SWARM beam states is

$$\mathbf{C} = \begin{bmatrix} (l_2 + l_3 + l_4) & (l_3 + l_4) & l_4 & 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

for the state vector $\mathbf{x} = \begin{pmatrix} \delta_1 & \delta_2 & \delta_3 & \dot{\delta}_1 & \dot{\delta}_2 & \dot{\delta}_3 \end{pmatrix}^T$. If the rate gyro is also measured, $\mathbf{y} = \begin{pmatrix} Y & \dot{\theta}_b \end{pmatrix}^T$, and the matrix becomes

$$\mathbf{C} = \begin{bmatrix} (l_2 + l_3 + l_4) & (l_3 + l_4) & l_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.15}$$

for the state vector $\mathbf{x} = \begin{pmatrix} \delta_1 & \delta_2 & \delta_3 & \dot{\theta}_b & \dot{\delta}_1 & \dot{\delta}_2 & \dot{\delta}_3 \end{pmatrix}^T$. Note that the state is only a subset of the full state vector defined in Section 2.2, implying that the measurements are only tied to a portion of the complete robot-beam state. This raises the important question of observability in the estimation problem: can any additional states be

estimated using the dynamic model of the beam, and is there sufficient information to estimate those states?

3.3.2 Observable States

With the linear measurement relationship defined, we would like to see if it is possible to extract state information from the measurements collected by the camera and the rate gyro. For a linear system, this is the question of observability: given the matrix pair $(\mathbf{A}, \mathbf{C})$ of the linear system

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned}$$

can the initial state $\mathbf{x}(0)$ (and therefore, through the known dynamics, all future states) be determined from a sequence of measurements $\mathbf{y}$? According to the well-known observability condition, for $\mathbf{A} \in \mathbb{R}^{nxn}$, the pair $(\mathbf{A}, \mathbf{C})$ is observable if and only if the observability matrix

$$\mathcal{O}_n = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

has rank n.

The linear dynamics for the assembly robot with state variables $\mathbf{x} = \begin{bmatrix} \mathbf{x} & \delta & \dot{\mathbf{x}} & \dot{\delta} \end{bmatrix}^T$ from Section 2.3 are repeated below:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{6x3} & \mathbf{0}_{6x3} & \mathbf{I}_{6x6} \\ \mathbf{0}_{6x3} & \mathbf{H} & \mathbf{0}_{6x6} \end{bmatrix}. \tag{3.16}$$

Which of these states are observable from beam deflection and gyro measurements? From the definition of observability, x_0, y_0, and θ_0 are unobservable because it is

impossible to determine the initial position and orientation of the robot by observing beam deflection and rotation rate. The global translation velocities $\dot{x}_0$ and $\dot{y}_0$ are similarly unavailable because the beam dynamics are the same regardless of the global translational velocity of the system. Both of these statements are evident by examining (3.16) and noticing that the block **0** matrices decouple the flexible dynamics from the global position and velocity states.[2] Since the states do not affect the flexible dynamics, an observable linear system for the estimator can be constructed by simply removing their rows and columns from the dynamic equations. It is important to note that while the beam dynamics are not coupled to the states $\mathbf{x}_0$ and $\dot{\mathbf{x}}_0$, these states *are* coupled to the beam dynamics through $\mathbf{H}$. The coupling allows us to observe the beam oscillation by measuring $\dot{\theta}_b$ through the rate gyro.

Removing the unobservable states, we are left with the following state vector, as conjectured in the previous section:

$$\mathbf{x} = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & \dot{\theta}_b & \dot{\delta}_1 & \dot{\delta}_2 & \dot{\delta}_3 \end{bmatrix}^T. \tag{3.17}$$

The resulting observable pair is $\mathbf{C}$ from Equation (3.15) and

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3x4} & \mathbf{I}_{3x3} \\ \mathbf{H}_{\theta\delta} & \mathbf{0}_{4x4} \end{bmatrix} \tag{3.18}$$

where $\mathbf{H}_{\theta\delta} \in \mathbb{R}^{4x3}$ is $\mathbf{H}$ with the appropriate rows removed. The beam model may also include viscous damping, which adds a matrix $\mathbf{D}$ to the lower right block:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3x4} & \mathbf{I}_{3x3} \\ \mathbf{H}_{\theta\delta} & \mathbf{D} \end{bmatrix}. \tag{3.19}$$

[2]This statement must be modified when the global velocity states have velocity-dependent damping. Though it would be difficult to accurately estimate the global velocity states through a viscous friction model, the base velocity would be coupled to the beam dynamics. For a space application, there will not be any global velocity damping, so we proceed by ignoring this effect.

3.3.3 Filter Equations

Now that we have developed observable state dynamics, the next step is to design a filter to extract state information from the measurements. Given a linear dynamic process corrupted by white Gaussian process noise w with $E\left[w_k w_k^T\right] = \mathbf{Q}_k$ and linear measurements of the states corrupted by white Gaussian sensor noise v with $E\left[v_k v_k^T\right] = \mathbf{R}_k$, the Kalman filter provides an optimal estimate of the system state. The Kalman filter is particularly useful in the context of assembly of flexible structures because it allows for time-varying dynamics. In principle, the matrix $\mathbf{A}_k$ could be adjusted on-line with an external parameter estimator, allowing for simultaneous system identification and estimation. The discrete form of the Kalman filter is summarized below:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k \tag{3.20}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{L}_k \left(\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_{k|k-1}\right) \tag{3.21}$$

$$\mathbf{L}_k = \mathbf{P}_{t|t-1} \mathbf{C}_k^T \left[\mathbf{C}_k \mathbf{P}_{k|k-1} \mathbf{C}_k^T + \mathbf{R}_k\right]^{-1} \tag{3.22}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{L}_k \mathbf{C}_k\right) \mathbf{P}_{k|k-1} \tag{3.23}$$

$$\mathbf{P}_{k+1|k} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{Q}_k \tag{3.24}$$

was implemented on the SWARM testbed, but the online parameter estimator remains as future work.

Equation (3.20) propagates the state forward using the estimator's internal model of the dynamics along with the commanded input to the system, and Equation (3.21) performs an update of the estimate using available measurements and the optimal gain calculated in Equation (3.22). The remaining equations update and propagate the error covariance of the state estimate. When the sequence of matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ is known, the time history of the gain and covariance is deterministic and reaches a stable steady-state. The steady-state gain can be used instead of the time-varying gain in many applications, but we retain the time varying form to allow for future system identification as described above.

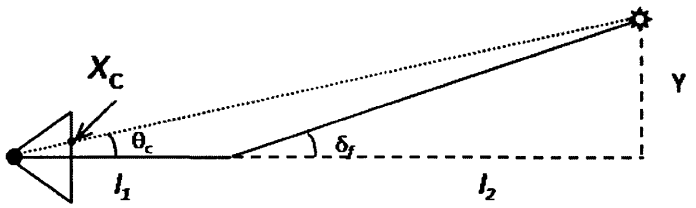


Figure 3.4: Simplified Beam Measurements

3.4 Simplified Beam Deflection Estimator

The simplified dynamic model developed in Section 2.3 has a reduced state vector

$$\mathbf{x} = \begin{bmatrix} x_0 & y_0 & \theta_0 & \delta_f & \dot{x}_0 & \dot{y}_0 & \dot{\theta}_0 & \dot{\delta}_f \end{bmatrix}^T. \tag{3.25}$$

. This representation greatly simplifies the estimation problem because the first three state variables and their derivatives are provided by the global estimator, and the beam deflection angle δ_f can be directly measured by the camera. The following sections develop an estimator for filtering and differentiating the deflection angle, δ_f, to provide the remaining state $\dot{\delta}_f$.

3.4.1 Measurement Equations

If the camera is not collocated with the flexible joint, a small transformation must be made to estimate the deflection angle. Starting with the perspective projection equations from (3.2),

$$\tan\theta_c = \frac{Y}{l_1 + l_2} = -\beta\frac{x_c}{f}. \tag{3.26}$$

Noting that the two triangles in Figure (3.4) share the beam deflection, Y, as a common side and substituting the equation above:

$$\begin{aligned} \tan\delta_f &= \frac{Y}{l_2} \\ &= \frac{l_1 + l_2}{l_2}\tan\theta_c \\ &= -\left(\frac{\beta}{f}\right)\left(\frac{l_1 + l_2}{l_2}\right)x_c. \end{aligned} \tag{3.27}$$

Assuming δ_f is small,

$$\delta_f \approx -\left(\frac{\beta}{f}\right)\left(\frac{l_1 + l_2}{l_2}\right) x_{c.} \tag{3.28}$$

Length l_1 represents the distance from the focal point of the camera to the flexible joint. l_2 is set to the length of the beam and assumed to be constant for small angles.

Equation (3.27) shows that the primary limit on angular resolution of the camera is the ratio of the pixel size, β, to the focal length, f. This result is intuitively correct because a camera with a larger focal length observes a narrower field of view for a the same number of pixels, and smaller pixel size implies that there are more pixels available to represent a given field of view. Although the deflection angle accuracy is not sensitive to the length of the beam, the deflection magnitude scales, to first order, with the length of the beam. Longer beams, therefore, require either a more tightly focused camera lens or a higher resolution camera to maintain the accuracy required for docking.

3.4.2 Linear Quadratic Estimator for Beam Angle

In section 3.3.3 a time varying implementation of the Kalman filter was used to retain the possibility of updating the beam dynamics through an external parameter estimator. Direct measurement of the beam deflection angle introduces a significant simplification because it is no longer necessary to embed the beam dynamic model in the estimator to extract the deflection angle δ_f. Without a need for updating the dynamics, a steady-state filter such as the Linear Quadratic Estimator (LQE) is a good choice for tracking the angle states.

For a full state estimate, it is still necessary to determine the angular velocity $\dot{\delta}_1$, which can be found by constructing an observer from the second order dynamics:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{3.29}$$

with state $\mathbf{x} = \begin{pmatrix} \delta_f & \dot{\delta}_f \end{pmatrix}^T$. The model assumes a constant velocity and lumps

the beam deflection dynamics into process noise. In the SWARM system, this is a reasonable assumption because the camera samples at approximately 25 Hz and large deflection beam dynamics are on the order of 0.1 Hz.. The discrete time model representation of the dynamics is

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k \tag{3.30}$$

with process noise $\mathbf{w}_k$. For the estimator output equation, Equation (3.28) is inverted to show camera measurement as a function of beam angle:

$$\begin{aligned} y_k &= x_c & (3.31) \\ &= \begin{bmatrix} -\left(\frac{f}{\beta}\right)\left(\frac{l_2}{l_1+l_2}\right) & 0 \end{bmatrix} \mathbf{x}_k + v_k & (3.32) \end{aligned}$$

with measurement noise, v_k.

The LQE uses a set of equations similar to the Kalman filter in 3.3.3, except that the innovations gain $\mathbf{L}_k$ is replaced with the steady-state gain $\mathbf{L}$ found by solving the well-known Discrete Algebraic Riccati Equation below

$$\begin{aligned} \mathbf{P}_{ss} &= \mathbf{Q}_k + \mathbf{A}_k \left(\mathbf{P}_{ss} - \mathbf{P}_{ss}\mathbf{C}^T \left[\mathbf{C}\mathbf{P}_{ss}\mathbf{C}^T + \mathbf{R} \right]^{-1} \mathbf{C}\mathbf{P} \right) \mathbf{A}_k^T & (3.33) \\ \mathbf{L} &= \mathbf{P}_{ss}\mathbf{C}^T\mathbf{R}^{-1} & (3.34) \end{aligned}$$

In steady-state, $\mathbf{L}$ is optimal for noise covariance characteristics specified by $E\left[\mathbf{w}_k\mathbf{w}_k^T\right] = \mathbf{Q}_k$ and $E\left[v_k^2\right] = R$. The remaining equations are the same as 3.20 and 3.21 with $\mathbf{B_k} = \mathbf{0}$ and $\mathbf{A}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$ for the discrete-time representation of (3.29).

Measurement	σ^2
x_{img}	$0.02\,pixel^2$
$\dot{\theta}_b$	$7.6x10^{-7}\left(\frac{rad}{s}\right)^2$

Table 3.5.1: Noise Variances for Gyro and Camera

3.5 Estimator Performance

3.5.1 Kalman Filter Simulation

This section will examine the performance of the full state filter under a set of increasingly challenging conditions while the system performs a representative assembly maneuver. For each test, the baseline nonlinear simulation was held constant with the same parameters as those used for the dynamic model simulations (see Table 2.4.1), except for a small amount of representative damping of $b = 0.005\frac{N \cdot m - s}{rad}$. The test maneuver consists of a simple rotation of 30 degrees under closed loop PD control, which was tuned to be slightly underdamped to examine the beam oscillations. None of the estimated joint variables are used for control so the maneuver is independent of the estimator performance. After 90 seconds the maneuver starts to repeat in the opposite direction to show the estimator response to a large deflection after convergence.

The camera system is modeled by applying the perspective projection equations to the simulated positions of the camera and LED, and Gaussian noise is added to the pixel measurements to corrupt the measurement. The rate gyro measurement from the rotation rate of the robot base is similarly augmented. Both noise variances were determined from measurements of the hardware sensors under static conditions and are listed in Table 3.5.1. Process noise is not explicitly added to the simulation though nonlinearities and the test conditions described below impose unmodeled disturbances. In each simulation the estimator is initialized with zero initial conditions and activated 5 seconds after the maneuver begins to demonstrate convergence properties.

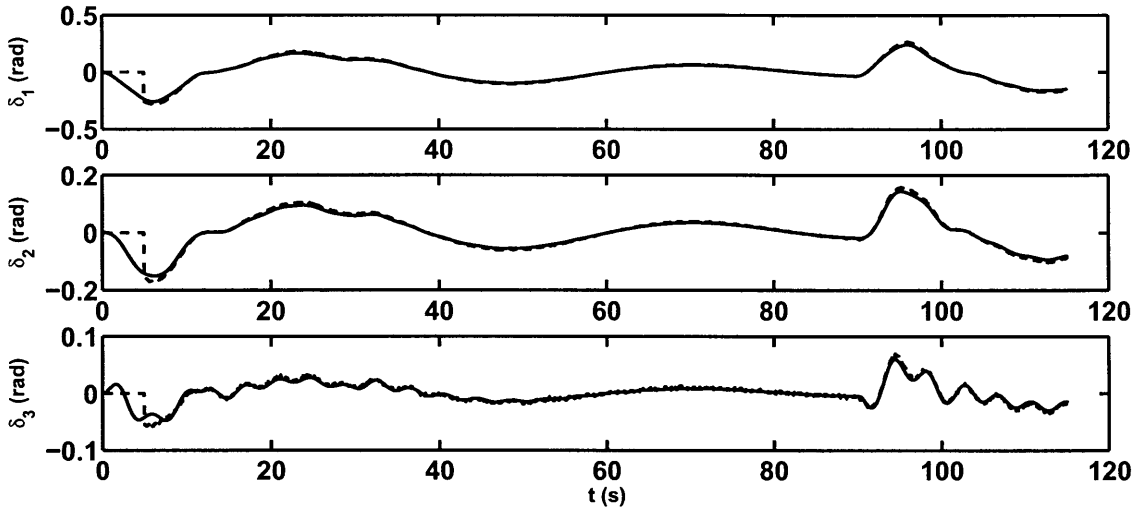


Figure 3.5: Deflection angle estimates (dashed lines) agree closely when the model is well known.

Nominal Case

For the first test, the estimator was configured with the true linearization of the beam dynamics using Equation (2.25) with the appropriate damping term. In this configuration we expect reasonably good performance because the open loop linear model is in close agreement with the nonlinear model. Figure 3.5 compares the nonlinear model to the estimated states, and as expected, the nominal case shows good performance even when the estimator starts with a large initial error. Likewise, the velocity states, shown in Figure 3.6, show a good estimate.

Uncertain Model

In this test, the estimator dynamics were configured with an incorrect linear model to test performance in the case when the true model is not exactly known. All spring constants were set to $k = 0.3\,\frac{N \cdot m}{rad}$, the damping was increased to $b = 0.01\frac{N \cdot m - s}{rad}$, and the mass of the air carriage and docking port at the end of the beam were lowered from $6.5\,kg$ to $4\,kg$. These modifications represent the propulsion tug docking to a new element of similar configuration but with slightly different physical properties. As Figure 3.7 shows, the velocity estimate is not quite as accurate as previous tests, particularly in representing higher frequency dynamics. Examining the velocity es-

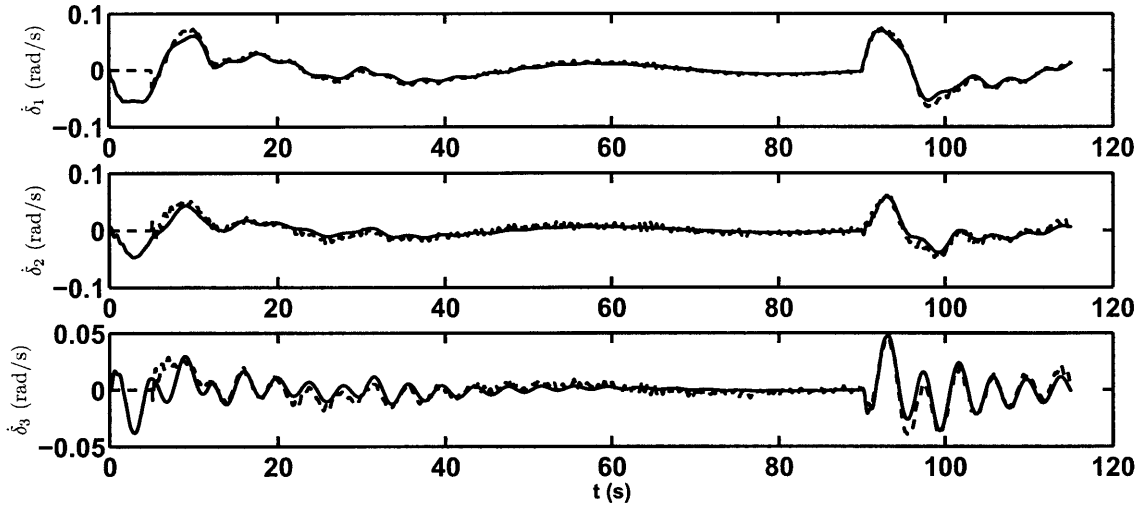


Figure 3.6: Velocity estimates (dashed lines) are also estimated accurately for the nominal model.

timates in Figure 3.8, the velocity estimate is rather poorly approximating the true state with a significant amount of noise. This sensitivity to model changes indicates that the estimator requires a very good dynamic model to provide accurate estimates of velocity.

Approximate Thrust Input Knowledge

Although it is frequently used for thruster reaction control, pulse width approximation is a source of error in the beam estimation system. This test simulates the uncertainty in the input caused by using a pulsed thruster system for approximating a continuous control signal. Thruster pulses are calculated by attempting to match the impulse that would have been delivered by the force command over a specified control period and are limited to a total percentage of the period to leave space for global ultrasound estimation. In Figure 3.9, an example thruster profile from a portion of the test maneuver are shown for a control period of 400ms with a duty cycle of 50 percent. The varying heights of the thruster pulses correspond to different numbers of active thrusters, with a single thruster providing approximately 0.006 $N \cdot m$ of torque.

In the simulation test, the estimator was configured to propagate the desired torque command from the controller, but thruster pulses were applied to the beam

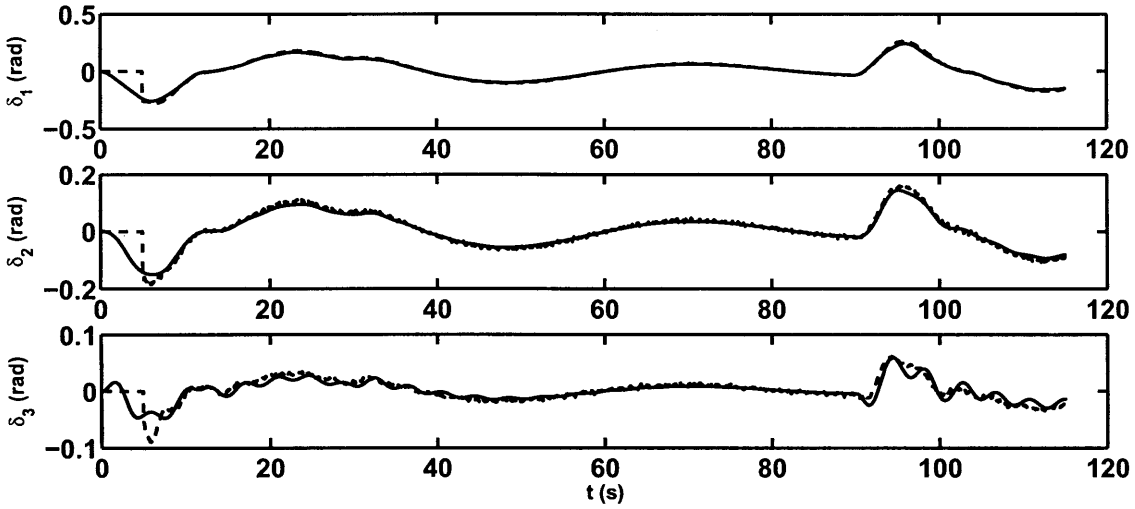


Figure 3.7: Angle estimates with the uncertain model remain close to their true values with some loss of higher frequency dynamics.

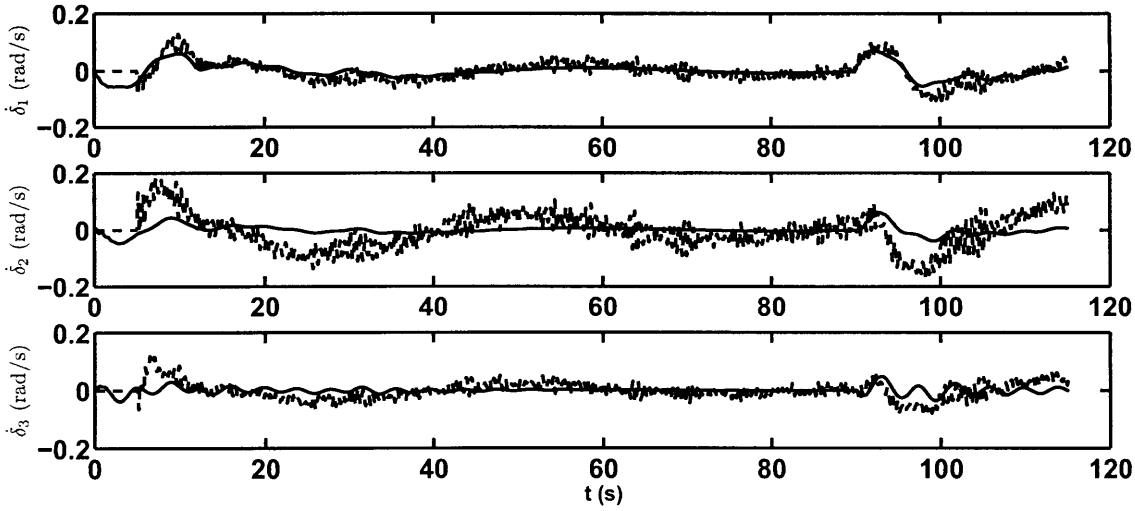


Figure 3.8: Velocity estimates for the uncertain model start to show significant errors.

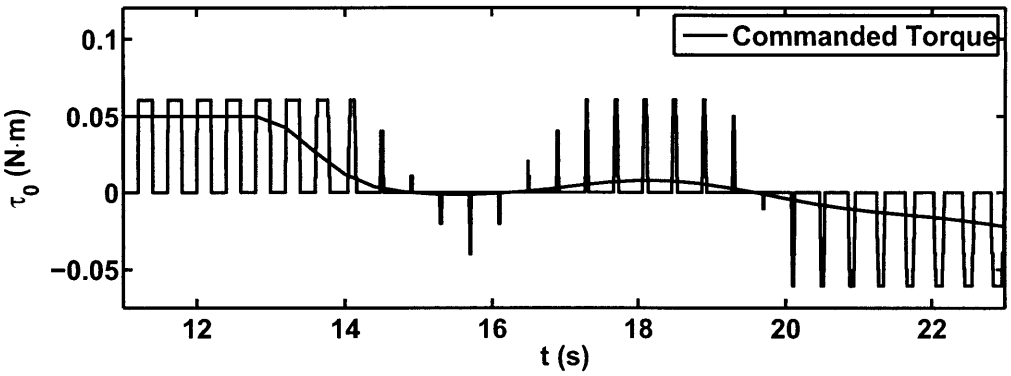


Figure 3.9: Example thruster pulses calculated from the commanded thrust profile.

dynamics. The nominal linear model was used in the estimator dynamics. Figures 3.10 and 3.11 show the resulting angle and angular velocity estimates. The true angle and angular velocity states show a high frequency oscillation due to the applied thruster pulses, but the tracking performance is reasonably accurate. The last frame of Figure 3.11 plots the desired torque command on the same axis as the third deflection angle. As shown by the flat regions in the control command, velocity errors are most prominent during high torque phases of maneuver when the commanded torque was intentionally saturated at $0.05\,N \cdot m$. The equivalent impulse for this command exceeds the maximum available firing time and leads to a mismatch between the actual thrust and the propagated thrust command. While thrust information contributes valuable information to the state estimate, it is important to provide accurate thrust values, especially when saturation effects are present. In addition, modeling errors, such as incorrect estimates of mass and inertia, can also cause large discrepancies when propagating expected thrust values.

3.5.2 Camera Calibration

3.5.3 LQE Testing

To compare the deflection angle estimator to the full state Kalman filter, the simulated rotation maneuver was repeated with the LQE observing the beam motion. Figure 3.12 compares the deflection angle and angular velocity estimate to the true value.

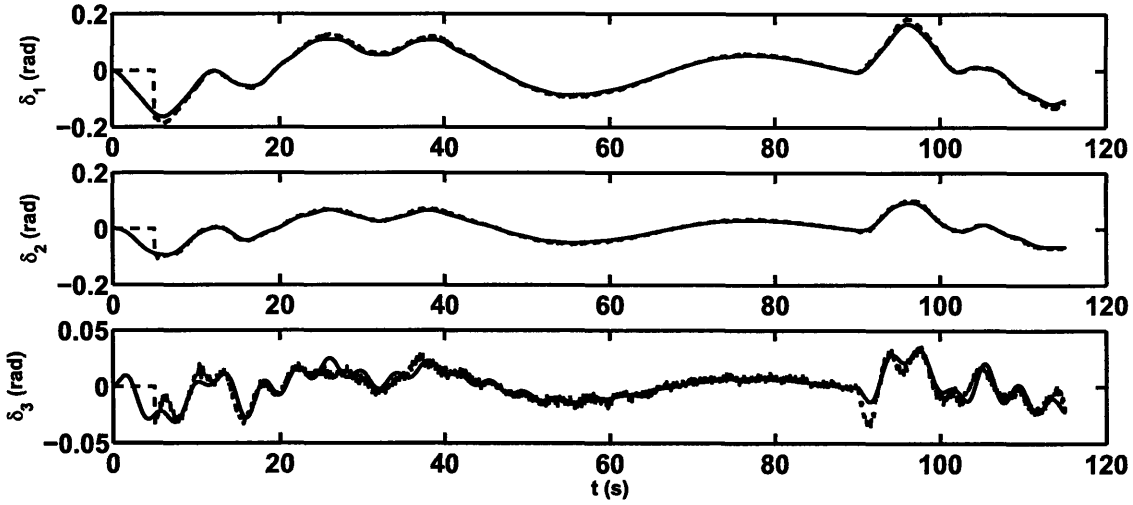


Figure 3.10: Angle estimates for the approximate thrust input simulation.

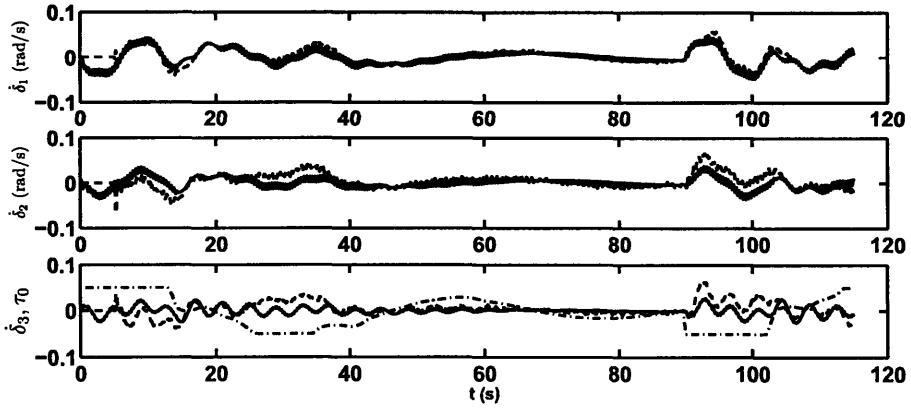


Figure 3.11: Velocity estimates show errors when the commanded thrust (dash-dot) saturates the thruster.

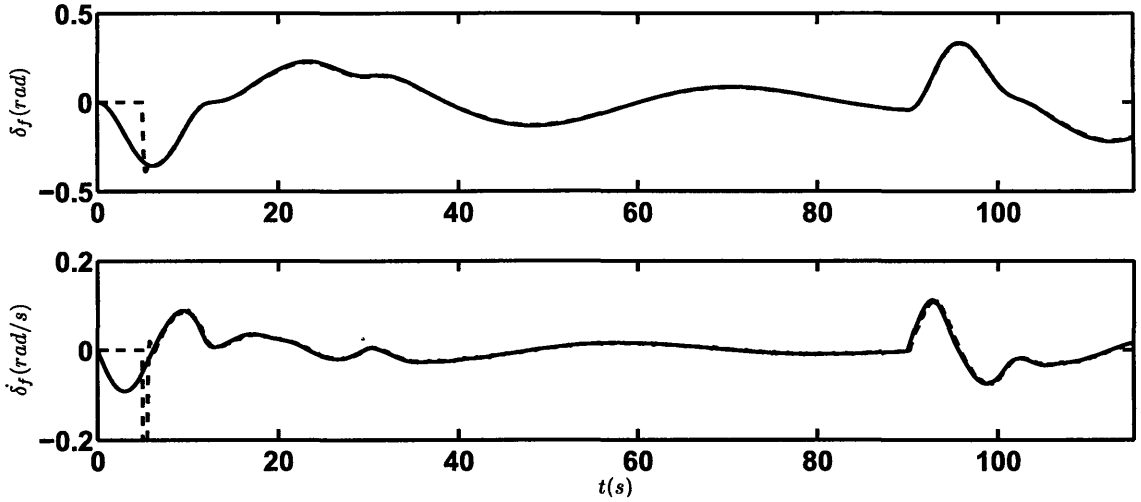


Figure 3.12: Deflection angle and velocity estimate (dashed) for the LQE estimator.

The deflection angle estimate tracks closely, even with deflection angles reaching 15 degrees. There is a large spike in the velocity estimate during the initial convergence period due to the relatively high filter gains and large initial error. With a short convergence period on the order of 0.5 seconds, the spike can be ignored as long as we require that no control should be active during the convergence period. An alternate restriction would be to initialize the filter with small initial deflections.

It is not necessary to compare the uncertain model case for deflection angle estimator because it does not use an internal model, but it is possible to examine the response for the pulsed thruster maneuver. As Figure 3.13 shows, the estimator is not sensitive to input uncertainty because it does not integrate thrust commands.

3.6 Conclusions

This chapter examined two estimator techniques for the flexible beam system: a full state Kalman filter, and a simplified deflection angle estimator. As a model-based estimator, the Kalman filter displays the best performance for cases where the dynamic model is well known. It has the important advantage of providing a more detailed estimate of the vibrational dynamics, including multiple vibrational modes. There are two major drawback to using the Kalman filter: first, as shown in simulation

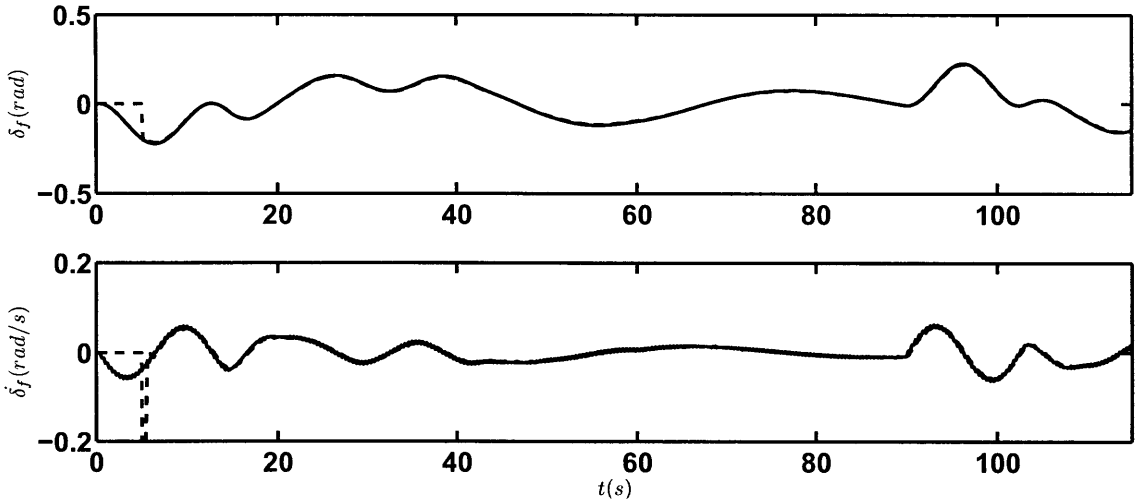


Figure 3.13: Deflection angle and velocity estimate (dashed) when the maneuver is performed with pulsed thrusters.

testing, uncertainties in the model degrade the accuracy of the estimate, and second, even if the model is well known, we must carry the dynamic model for every potential system to be manipulated. For an assembly task with small but significant variations between elements this approach may become infeasible. One solution to this problem is to perform online system identification in parallel with estimation, though this remains as future work.

As with the simplified dynamic model, the deflection angle estimator is desirable for on-orbit assembly because it has a simple parameterization. For observing planar oscillation, only the length of the beam is necessary for obtaining a very accurate estimate of the deflection angle and its derivative, and this information is sufficient to provide full state feedback for the simplified dynamic model. The compact representation comes at the cost of ignoring some of the vibrational dynamics at the level of the estimator. Note that the deflection angle is another form of the beam deflection measurement provided to the Kalman filter, and as shown in in the estimator derivation, contains information about the full state. Depending on the form of the controller, this information may still be used to eliminate vibration.

Although we have examined the performance of both estimators in the context of a simple closed-loop maneuver, they will be best evaluated in concert with the control

techniques developed in the following chapter.

Chapter 4

Flexible Beam Control for Assembly

4.1 Overview

With a set of dynamic models from Chapter 2, and estimators from Chapter 3, we are ready to tackle the task of maneuvering a flexible structure. This chapter examines three types of controllers that take different approaches to trajectory tracking and stabilization. The first case is a simple PD controller, which will serve as a baseline for comparison with the other two controllers. The second controller is a Partial Feedback Linearization (PFL) controller, commonly used for underactuated robots, and the final controller is a nonlinear adaptive controller which will be implemented to compensate for model uncertainty.

4.2 PD Control

PD control is among the simplest and most widely used techniques used for controlling dynamical systems. For planar rigid robot manipulators, PD setpoint control has been shown to provide global asymptotic stability [26], and this result has been extended to multi-link flexible manipulators [32]. In this section, stability is shown for a PD setpoint controller operating only on the actuated states of the segmented beam system. This is a type of *collocated* controller because the controlled outputs are directly measured.

4.2.1 Damped Case

Starting with the partitioned dynamics from (2.20), we initially assume that the system also includes a viscous damping term, $\mathbf{D}\dot{\mathbf{q}}$,

$$\begin{bmatrix} \mathbf{M}_{xx} & \mathbf{M}_{x\delta} \\ \mathbf{M}_{\delta x} & \mathbf{M}_{\delta\delta} \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{x}} \\ \ddot{\boldsymbol{\delta}} \end{pmatrix} + \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{x\delta} \\ \mathbf{C}_{\delta x} & \mathbf{C}_{\delta\delta} \end{bmatrix} \begin{pmatrix} \dot{\mathbf{x}} \\ \dot{\boldsymbol{\delta}} \end{pmatrix} + \mathbf{D}\dot{\mathbf{q}} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_f \end{pmatrix} \mathbf{q} = \begin{pmatrix} \boldsymbol{\tau}_x \\ \mathbf{0} \end{pmatrix}. \tag{4.1}$$

This system is open loop stable, but we wish to show that the system is stable under PD control. The input torque τ_x is determined by a setpoint PD controller for the desired state $\mathbf{q}_{a,d}$

$$\tau_a = -\mathbf{K}_p\tilde{\mathbf{x}} - \mathbf{K}_d\dot{\mathbf{x}} \tag{4.2}$$

where the error between the current position and the desired setpoint is $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_d$. $\mathbf{K}_p$ and $\mathbf{K}_d$ are selected as positive definite proportional and derivative gain matrices. To show the stability of this control law, the following energy-like Lyapunov function is proposed

$$V = \frac{1}{2}\left[\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} + \boldsymbol{\delta}^T\mathbf{K}_f\boldsymbol{\delta} + \tilde{\mathbf{x}}^T\mathbf{K}_p\tilde{\mathbf{x}}\right]. \tag{4.3}$$

The first two terms are the kinetic and potential energy of the system respectively, and the last term is the virtual potential energy stored in the proportional term of the control law. The time-derivative of V is

$$\begin{aligned} \dot{V} &= \dot{\mathbf{q}}^T\mathbf{M}\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T\dot{\mathbf{M}}\dot{\mathbf{q}} + \dot{\boldsymbol{\delta}}^T\mathbf{K}_f\boldsymbol{\delta} + \dot{\mathbf{x}}^T\mathbf{K}_p\tilde{\mathbf{x}} \\ &= \dot{\mathbf{q}}^T\left[\begin{pmatrix} \boldsymbol{\tau}_x \\ \mathbf{0} \end{pmatrix} - (\mathbf{C}+\mathbf{D})\dot{\mathbf{q}} - \begin{pmatrix} \mathbf{0} \\ \mathbf{K}_f\boldsymbol{\delta} \end{pmatrix}\right] + \\ &\quad \frac{1}{2}\dot{\mathbf{q}}^T\dot{\mathbf{M}}\dot{\mathbf{q}} + \dot{\boldsymbol{\delta}}^T\mathbf{K}_f\boldsymbol{\delta} + \dot{\mathbf{x}}^T\mathbf{K}_p\tilde{\mathbf{x}}. \end{aligned} \tag{4.4}$$

Applying the skew symmetry property from (2.5), $\dot{\mathbf{q}}^T\left(\dot{\mathbf{M}} - 2\mathbf{C}\right)\dot{\mathbf{q}} = 0$, and substituting the control law from equation (4.2),

$$\dot{V} = \left(-\dot{\boldsymbol{\delta}}^T\mathbf{K}_f\boldsymbol{\delta} - \dot{\mathbf{x}}^T\mathbf{K}_p\tilde{\mathbf{x}} - \dot{\mathbf{x}}^T\mathbf{K}_d\dot{\mathbf{x}} - \dot{\mathbf{q}}^T\mathbf{D}\dot{\mathbf{q}}\right) + \dot{\boldsymbol{\delta}}^T\mathbf{K}_f\boldsymbol{\delta} + \dot{\mathbf{x}}^T\mathbf{K}_p\tilde{\mathbf{x}}. \tag{4.5}$$

Most of the remaining terms cancel, leaving

$$\dot{V} = -\dot{\mathbf{x}}^T \mathbf{K}_d \dot{\mathbf{x}} - \dot{\mathbf{q}}^T \mathbf{D} \dot{\mathbf{q}} \leq 0 \tag{4.6}$$

To show that this results in global stability, we must examine the dynamics in the case of $\dot{V} = 0$. Since $\mathbf{K}_d$ and $\mathbf{D}$ are assumed to be positive definite, $\dot{V} = 0$ implies

$$\dot{\mathbf{q}} = \mathbf{0}. \tag{4.7}$$

Substituting this into the robot dynamics

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1} \begin{pmatrix} -\mathbf{K}_p (\mathbf{x} - \mathbf{x}_d) \\ -\mathbf{K}_f \boldsymbol{\delta} \end{pmatrix} \tag{4.8}$$

Since $\mathbf{M}$, $\mathbf{K}_p$, and $\mathbf{K}_f$ are all positive definite, the only steady state for which $\dot{V} = 0$ is $\mathbf{x} = \mathbf{x}_d$ and $\boldsymbol{\delta} = \mathbf{0}$.

4.2.2 Undamped Case

The above result depends on the presence of damping in at least the flexible dynamics to show asymptotic convergence. For operations in a laboratory this is a reasonable assumption, but structures in space may be very lightly damped. The stability of a PD control law for an undamped flexible beam in a gravity field was shown in [5] and the result is extended here for an undamped segmented flexible beam.

If we assume that the kinetic energy of the system is represented by the beam in its undeformed state ($\boldsymbol{\delta} = \mathbf{0}$), then we can consider the matrix $\mathbf{M}$ as a function of $\mathbf{x}$ only, and it has the structure

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{xx}(\mathbf{x}) & \mathbf{M}_{x\delta}(\mathbf{x}) \\ \mathbf{M}_{\delta x}(\mathbf{x}) & \mathbf{M}_{\delta\delta} \end{bmatrix} \tag{4.9}$$

An important result of the assumption is that the lower right submatrix $\mathbf{M}_{\delta\delta}$ is constant. Keeping the definition of $\mathbf{C}$ from (2.4), the $\mathbf{M}$ and $\mathbf{C}$ maintain their skew-

symmetry property, but now $\mathbf{C} = \mathbf{0}$ whenever $\dot{\mathbf{x}} = \mathbf{0}$.

Repeating the procedure in the previous section without the damping term results in

$$\dot{V} = -\dot{\mathbf{x}}^T \mathbf{K}_d \dot{\mathbf{x}} \leq 0 \tag{4.10}$$

Now $\dot{V} = 0$ implies only that

$$\dot{\mathbf{x}} = \ddot{\mathbf{x}} = \mathbf{0}. \tag{4.11}$$

Substituting this condition into the dynamics,

$$\mathbf{M}_{x\delta}(\mathbf{x})\ddot{\boldsymbol{\delta}} = -\mathbf{K}_p(\mathbf{x} - \mathbf{x}_d) \tag{4.12}$$

$$\mathbf{M}_{\delta\delta}\ddot{\boldsymbol{\delta}} + \mathbf{K}_f\boldsymbol{\delta} = \mathbf{0}. \tag{4.13}$$

The set of coupled, constant coefficient, linear, homogeneous equations (4.13) suggest that $\boldsymbol{\delta}$ has a solution that is a function of time, which we can substitute in (4.12):

$$-\mathbf{M}_{x\delta}(\mathbf{x})\mathbf{M}_{\delta\delta}^{-1}\mathbf{K}_f\ \boldsymbol{\delta}(t) = -\mathbf{K}_p(\mathbf{x} - \mathbf{x}_d). \tag{4.14}$$

Due to equation (4.11), the right side of this equation is constant, but the left hand side is also constant. In [5], De Luca shows that equation (4.14) is a contradiction because it equates the sum of independent time-varying functions with a constant. The equation can only be satisfied if $\boldsymbol{\delta}(t) = 0$ and $\mathbf{x} = \mathbf{x}_d$. With this result, we conclude that the setpoint error and the joint deflections are globally asymptotically stable without damping.

4.2.3 Discussion

The previous two sections suggest that collocated PD control is globally asymptotically stable for any choice of gains $\mathbf{K}_d$ and $\mathbf{K}_p$. In practice, however, additional unmodeled effects such as time delays and discretization can easily introduce instabilities. Furthermore, the stability results only imply asymptotic stability in the limit

of $t \rightarrow \infty$, and do not provide direct results about the convergence rate. As noted by Talebi et al. in [28], collocated PD control only weakly affects the flexible dynamics, resulting in long convergence times. It is also important to note that the results above apply only to fixed setpoints. Tracking control imposes more serious limitations on the simultaneous convergence of the flexible and rigid dynamics, which will be examined further in the next section. With this in mind, we use PD control as it is often used in the literature, as a benchmark or as a stabilizing component of a more complicated control law.

4.3 Partial Feedback Linearization Control

Feedback linearizing control laws make use of a nonlinear control feedback term to cause the dynamics of a nonlinear system to follow the behavior of a simple linear system. Underactuated systems, such as the Acrobot [27], inverted pendula, or flexible manipulators, are not feedback linearizable, but it is possible to perform a similar approach such that a subset of the dynamics are linearized by feedback. This approach is referred to as Partial Feedback Linearization (PFL) control and has been applied successfully to a number of classic underactuated problems [27]. This section will develop a PFL controller for the flexible beam maneuvering problem.

4.3.1 Partial Feedback Linearization for Beam Maneuvering

Starting with the partitioned nonlinear dynamics, we will treat the actuated and unactuated parts as separate equations

$$\mathbf{M}_{xx}\ddot{\mathbf{x}} + \mathbf{M}_{x\delta}\ddot{\boldsymbol{\delta}} + \mathbf{C}_x(\dot{\mathbf{q}}) = \boldsymbol{\tau}_x \tag{4.15}$$

$$\mathbf{M}_{\delta x}\ddot{\mathbf{x}} + \mathbf{M}_{\delta\delta}\ddot{\boldsymbol{\delta}} + \mathbf{C}_\delta(\dot{\mathbf{q}}) + \mathbf{K}_f\boldsymbol{\delta} = \mathbf{0}. \tag{4.16}$$

The subset of the state to be feedback linearized, also referred to as the *task space*, is defined by an output function of the state

$$\mathbf{y} = \mathbf{f}(\mathbf{q}) \tag{4.17}$$

where $\mathbf{y} \in \mathbb{R}^m$ with $m \leq \dim(\tau_x)$. The objective of the PFL controller is to design τ_x as a function of a new input $\mathbf{v}$ in such a way that the dynamics of $\mathbf{y}$ are simply

$$\ddot{\mathbf{y}} = \mathbf{v}. \tag{4.18}$$

To find the appropriate control law, we start by differentiating the output equation twice,

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{J}\dot{\mathbf{q}} \\ \ddot{\mathbf{y}} &= \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}_x\ddot{\mathbf{x}} + \mathbf{J}_\delta\ddot{\boldsymbol{\delta}}. \end{aligned} \tag{4.19}$$

The Jacobian of the task space, $\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}$, is partitioned in the same way as the dynamics, into actuated and unactuated components:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_x & \mathbf{J}_\delta \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{y}}{\partial \mathbf{x}} & \frac{\partial \mathbf{y}}{\partial \delta} \end{bmatrix}. \tag{4.20}$$

We also define the generalized Jacobian

$$\bar{\mathbf{J}} = \mathbf{J}_x - \mathbf{J}_\delta \mathbf{M}_{\delta\delta}^{-1}\mathbf{M}_{\delta x}. \tag{4.21}$$

Solving the unactuated dynamics in Equation (4.16) for $\ddot{\boldsymbol{\delta}}$,

$$\ddot{\boldsymbol{\delta}} = -\mathbf{M}_{\delta\delta}^{-1}\left(\mathbf{M}_{\delta x}\ddot{\mathbf{x}} + \mathbf{C}_\delta(\dot{\mathbf{q}}) + \mathbf{K}_f\boldsymbol{\delta}\right) \tag{4.22}$$

and substituting into Equation 4.19

$$\begin{aligned}\ddot{\mathbf{y}} &= \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}_x\ddot{\mathbf{x}} - \mathbf{J}_\delta\mathbf{M}_{\delta\delta}^{-1}\left(\mathbf{M}_{\delta x}\ddot{\mathbf{x}} + \mathbf{C}_\delta(\dot{\mathbf{q}}) + \mathbf{K}_f\boldsymbol{\delta}\right) \\ &= \dot{\mathbf{J}}\dot{\mathbf{q}} + \bar{\mathbf{J}}\ddot{\mathbf{x}} - \mathbf{J}_\delta\mathbf{M}_{\delta\delta}^{-1}\left(\mathbf{C}_\delta(\dot{\mathbf{q}}) + \mathbf{K}_f\boldsymbol{\delta}\right). \end{aligned} \tag{4.23}$$

Suppose a controller is selected such that the actuated dynamics to satisfy

$$\ddot{\mathbf{x}} = \bar{\mathbf{J}}^+\left[\mathbf{v} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}_\delta\mathbf{M}_{\delta\delta}^{-1}\left(\mathbf{C}_\delta(\dot{\mathbf{q}}) + \mathbf{K}_f\delta\right)\right] \tag{4.24}$$

where $\bar{\mathbf{J}}^+$ is the right pseudo-inverse

$$\bar{\mathbf{J}}^+ = \bar{\mathbf{J}}^T\left(\bar{\mathbf{J}}\bar{\mathbf{J}}^T\right)^{-1} \tag{4.25}$$

subject to the rank condition

$$rank\left(\bar{\mathbf{J}}\right) = p. \tag{4.26}$$

Substituting Equation (4.24) into Equation (4.23) results in

$$\ddot{\mathbf{y}} = \mathbf{v}.$$

The new control input $\mathbf{v}$ can be calculated using a simple linear control law, such as a PD controller with trajectory acceleration feed-forward,

$$\mathbf{v} = \ddot{\mathbf{y}}_d - \mathbf{K}_d\dot{\tilde{\mathbf{y}}} - \mathbf{K}_p\tilde{\mathbf{y}} \tag{4.27}$$

where $\tilde{(\bullet)} = (\bullet) - (\bullet)_d$ is the error between an output and the desired trajectory. Subtracting the right side results in the error dynamics

$$\ddot{\tilde{\mathbf{y}}} + \mathbf{K}_d\dot{\tilde{y}} + \mathbf{K}_p\tilde{\mathbf{y}} = \mathbf{0}. \tag{4.28}$$

If the gain matrices $\mathbf{K}_p$ and $\mathbf{K}_d$, are selected to make equation (4.28) stable, the trajectory error will be stable. The only remaining step is to extract $\boldsymbol{\tau}_x$, from $\mathbf{v}$, which is achieved by substituting Equation (4.24) into Equations (4.22) and (4.15)

and equating the actuator command to Equation (4.15).

The preceding derivation shows that a control solution can be selected to achieve perfect tracking of the selected output equation as long as the the rank condition in (4.26) is satisfied. Unfortunately, perfect tracking of the output equation does not guarantee that the entire system is closed-loop stable and stability is closely tied to the choice of the task space.

4.3.2 Choice of Task

Shkolnik and Tedrake note that the task space can be thought of as a fully actuated template dynamical system that is embedded in the higher dimensional underactuated dynamics [23]. For the swing-up problem of a multi-link inverted pendulum with a single actuator, the template system might consist of a rigid rod that rotates from the hanging to the upright position. In this case, the single actuator controls one degree of freedom, the angular position of the rod. With more actuators available, the template model may contain more degrees of freedom, such as angular position and length.

For an assembly operation, the objective is to move a flexible structure with minimal excitation. The task space can consist of, at most, 3 dimensions, and this restriction poses the subtle challenge of finding a representation that simultaneously captures the 3D rigid body state and the desire to prevent oscillations. One candidate template model is simply a rigid beam for which reference trajectories are selected as combination of rigid body rotations and translations. This representation is a *collocated* task space because the output variables are the same as the actuated states, and therefore

$$\mathbf{y} = \mathbf{x}. \tag{4.29}$$

To control vibrations, we might instead select the unactuated, or *non-collocated,* states as the outputs

$$\mathbf{y} = \boldsymbol{\delta}. \tag{4.30}$$

Both outputs satisfy the conditions from the previous section for achieving perfect

tracking, but neither accomplish the goals for maneuvering the flexible beam. To see why, consider the behavior of the system after the output converges to the desired trajectory. For the collocated case, the beam position and rotation will be constrained to their targets, but the beam may still oscillate freely. For the non-collocated case, the beam oscillations can be eliminated, but the global position is free to drift. Clearly, the task space must include elements of both output equations, but it is not immediately clear how to combine them.

The preceding examples are related to the concept of zero dynamics, the remaining dynamic behavior when the output variables are held at zero. For a nonlinear system, asymptotically stable zero dynamics imply that the system is minimum phase, otherwise the system is non-minimum phase. Most studies of feedback linearization applied to flexible manipulators focus on the selection of a set of outputs that achieve minimum phase behavior and therefore result in global stability. The key for determining the task space for the robot-beam system will be choosing a set of outputs that is minimum phase.

The first two outputs must be used to constrain the x and y positions of the robot base in the workspace, and the remaining output will be used for orientation and beam control. Arisoy et al. consider switching back and forth between Equations (4.29) and (4.30) to damp the beam after the trajectory has reached its target [1], but the objective of this study is to develop a controller that is also capable of attenuating oscillations while following a trajectory. The approach used here is similar to the one presented by Talebi et al. in [28], where the output is defined as a weighted combination of the base rotation angle, θ_b, and the beam deflection angle, δ_f, calculated from Equation (2.31) or taken directly from the simplified beam model

$$y_{beam} = \theta_b + \alpha\delta_f. \tag{4.31}$$

The weighting factor, $-1 < \alpha < 1$, can be viewed as a redefinition of the tip position to another location on the beam. For small values of α, the tip is relocated near the base, and for negative values of α, the tip is reflected about the length of the beam. As

shown in [28], there exists a critical value, α^*, such that all outputs for $-1 < \alpha < \alpha^*$ are minimum phase, provided that the system has at least some damping of the flexible dynamics.

As an example of how this modification affects system dynamics, consider the transfer function from the base torque input, $u = \tau_b$ to the total angular beam deflection $y = \theta_b + \delta_f$ in the linearized dynamics. Using a representative linearization from the SWARM hardware with beam damping (see Equation (2.27) and Table 2.4.1), the transfer function is

$$\frac{Y(s)}{U(s)} = \frac{-0.3989s^6 - 0.742s^5 - 29.72s^4 - 3.053s^3 - 61.05s^2 + 0.325s + 4.334}{s^8 + 2.578s^7 + 77.14s^6 + 61.07s^5 + 209.7s^4 + 119.4s^3 + 48.72s^2 + 4.334s} \tag{4.32}$$

for which the pole-zero map is shown in Figure 4.1a. This system is non-minimum phase, as evidenced by the zero in the right half-plane. If instead, we consider the transfer function from $u = \tau_b$ to the augmented output $y = \theta_b + \alpha\delta_f$, then the transfer function is

$$\frac{Y(s)}{U(s)} = \frac{0.1427s^6 + 0.2641s^5 + 10.58s^4 + 1.368s^3 + 27.36s^2 + 0.325s + 4.334}{s^8 + 2.578s^7 + 77.14s^6 + 61.07s^5 + 209.7s^4 + 119.4s^3 + 48.72s^2 + 4.334s} \tag{4.33}$$

for $\alpha = 0.5$. The pole-zero map shown in Figure 4.1b shows that the transfer function for the augmented output is minimum phase because there are no zeros in the right half-plane. The process of selecting an appropriate α and extending the minimum phase result to the full nonlinear dynamics can be found in [28]. We assume for the remainder of the discussion that an appropriate value of α is selected. Supplying a base orientation trajectory for y_{beam} results in asymptotic tracking by the combined output, and after the target is reached, the beam dynamics should decay due to the stable zero-dynamics.

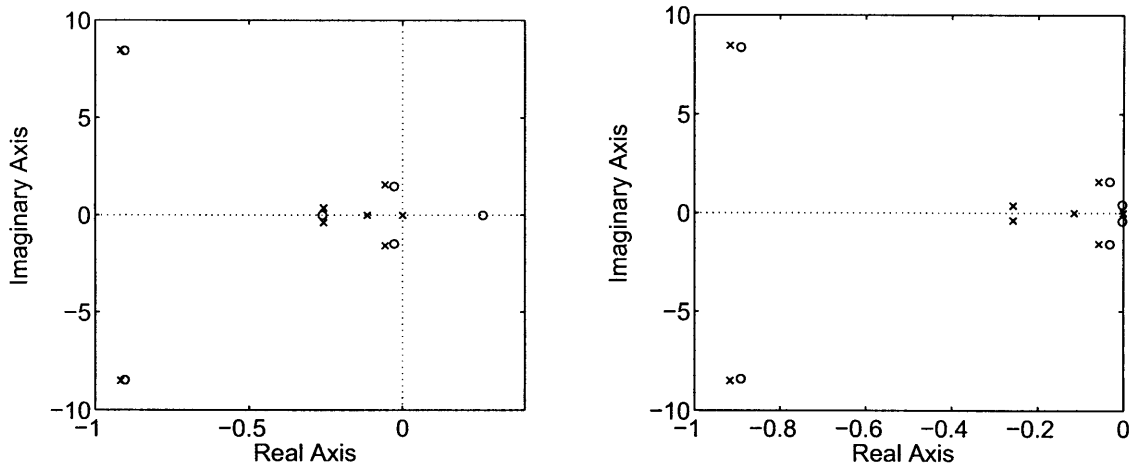


Figure 4.1: Transfer function (a) τ_b to $\theta_b + \delta_f$ is nonminimum phase, but (b) τ_b to $\theta_b + 0.5\delta_f$ is minimum phase.

4.3.3 Discussion

To summarize the PFL control law,

$$\mathbf{y} = \begin{bmatrix} x_b \\ y_b \\ \theta_b + \alpha\delta_f \end{bmatrix} \tag{4.34}$$

defines the task space as the Cartesian base positions and a weighted deflection angle measurement, and

$$\mathbf{v} = \ddot{\mathbf{y}}_d - \mathbf{K}_d\dot{\tilde{\mathbf{y}}} - \mathbf{K}_p\tilde{\mathbf{y}} \tag{4.35}$$

defines the input to the feedback linearized subsystem.

$$\ddot{\mathbf{x}} = \bar{\mathbf{J}}^{+}\left[\mathbf{v} - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{J}_\delta\mathbf{M}_{\delta\delta}^{-1}\left(\mathbf{C}_\delta(\dot{\mathbf{q}}) + \mathbf{K}_f\boldsymbol{\delta}\right)\right] \tag{4.36}$$

and

$$\ddot{\boldsymbol{\delta}} = -\mathbf{M}_{\delta\delta}^{-1}\left(\mathbf{M}_{\delta x}\ddot{\mathbf{x}} + \mathbf{C}_\delta(\dot{\mathbf{q}}) + \mathbf{K}_f\boldsymbol{\delta}\right) \tag{4.37}$$

are used to calculate the actuator command using

$$\tau_x = \mathbf{M}_{xx}\ddot{\mathbf{x}} + \mathbf{M}_{x\delta}\ddot{\boldsymbol{\delta}} + \mathbf{C}_x(\dot{\mathbf{q}}). \tag{4.38}$$

PFL control represents a significant improvement over the PD control law developed in Section 4.2 because it explicitly incorporates stable trajectory tracking and provides improved damping of the flexible beam. The added performance can be attributed to utilizing *a priori* knowledge of the system dynamics from the embedded model in the control law. Unfortunately, the model must be known, and as shown in the simulation results in Section (4.5), uncertainties in the dynamic model can have a significant effect on control performance. In the context of an assembly operation, the PFL controller requires a thoroughly identified dynamic model for every unique structural element. Furthermore, the controller assumes the availability of the full state ($\mathbf{q}$, $\dot{\mathbf{q}}$), which requires a state estimator with its own dynamic model for each element. It may be possible to find common parameterizations between the the control and estimation models as well as approximations like the simplified beam model to reduce complexity, but the problem of obtaining an accurate model remains. Therefore, we will consider PFL as a stepping stone in the design of a more robust and practical control system. In particular, the design of the PFL controller offers the following insights:

Task Space Tradeoff An underactuated controller can only exactly track a subspace of the full state of the robot, limited in size by the number of actuators. This lesson is helpful for designing other controllers that may not explicitly state this restriction.

Output Selection The outputs for tracking control must be carefully selected to ensure minimum phase behavior. To achieve a minimum phase system, we lose perfect tracking ability of the beam tip position but gain the ability to damp out vibrations quickly. Setpoint control of the tip position is still possible by rotating the base and damping out residual oscillations.

4.4 Adaptive Control

In this section an adaptive controller is constructed for the control of a flexible beam. Traditional adaptive control for robot manipulators requires a fully actuated dynamic model, which is reviewed in the first section. The second section covers a modification of the fully actuated model called *normal form augmentation* [9], which allows for underactuated control under restrictions similar to the PFL controller.

4.4.1 Review of Adaptive Control for Fully Actuated Systems

The following is a summary of the adaptive trajectory control for robotic manipulators presented in [26] and applies to many systems with the dynamics of the form shown in Equation (2.3) and below:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}. \tag{4.39}$$

Here, $\mathbf{q} \in \mathbb{R}^n$, and unlike in previous sections, we assume the control input, τ, has no zero elements. The control law makes use of a sliding variable, $\mathbf{s} \in \mathbb{R}^n$, which is the weighted sum of velocity and position errors

$$\mathbf{s} = \dot{\tilde{\mathbf{q}}} + \boldsymbol{\Lambda}\tilde{\mathbf{q}} \tag{4.40}$$

with $\tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}_d$ and $\boldsymbol{\Lambda}$is a positive definite weighting matrix. The objective of the controller is to constrain the dynamics to the surface $\mathbf{s} = \mathbf{0}$, after which the tracking error converges with the behavior of a first order linear system. Rearranging Equation (4.40), $\mathbf{s}$ can also be interpreted as a velocity error

$$\mathbf{s} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_r \tag{4.41}$$

$$\dot{\mathbf{q}}_r = \dot{\mathbf{q}}_d - \boldsymbol{\Lambda}\tilde{\mathbf{q}} \tag{4.42}$$

where $\dot{\mathbf{q}}_r$ represents a special reference velocity weighted by position tracking error.

The dynamics for any mechanical system depend linearly on a set of constant

physical parameters, $\mathbf{a} \in \mathbb{R}^p$, that represent combinations of masses, inertias and link lengths. Using this property, the left hand side of Equation (4.39) can be rewritten as

$$\mathbf{Ya} = \mathbf{M(q)\ddot{q}} + \mathbf{C(q,\dot{q})\dot{q}} + \mathbf{G(q)}. \tag{4.43}$$

The matrix $\mathbf{Y} \in \mathbb{R}^{nxp}$ is known as a *dynamic regressor* and contains the nonlinear terms of the dynamics. Details on calculating the dynamic regressor can be found in [17].

To derive the adaptive control law, we start with the following Lyapunov function

$$V = \frac{1}{2}\left[\mathbf{s}^T\mathbf{M}\mathbf{s} + \tilde{\mathbf{a}}^T\mathbf{\Gamma}^{-1}\tilde{\mathbf{a}}\right] \tag{4.44}$$

with $\tilde{\mathbf{a}} = \hat{\mathbf{a}} - \mathbf{a}$. Differentiating with respect to time

$$\begin{aligned} \dot{V} &= \mathbf{s}^T\mathbf{M}\dot{\mathbf{s}} - \frac{1}{2}\mathbf{s}^T\dot{\mathbf{M}}\mathbf{s} + \dot{\hat{\mathbf{a}}}^T\mathbf{\Gamma}^{-1}\tilde{\mathbf{a}} \\ &= \mathbf{s}^T\mathbf{M}\left(\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_r\right) - \frac{1}{2}\mathbf{s}^T\dot{\mathbf{M}}\mathbf{s}. \end{aligned} \tag{4.45}$$

Substituting the dynamics from (4.39)

$$\dot{V} = \mathbf{s}^T\left(\boldsymbol{\tau} - -\mathbf{M}\ddot{\mathbf{q}}_\mathbf{r} - \mathbf{C}\dot{\mathbf{q}} - \mathbf{G}\right) - \frac{1}{2}\mathbf{s}^T\dot{\mathbf{M}}\mathbf{s} + \dot{\hat{\mathbf{a}}}^T\mathbf{\Gamma}^{-1}\tilde{\mathbf{a}}. \tag{4.46}$$

Using $\dot{\mathbf{q}} = \mathbf{s} + \dot{\mathbf{q}}_r$ and the skew-symmetry property

$$\dot{V} = \mathbf{s}^T\left(\boldsymbol{\tau} - \mathbf{M}\ddot{\mathbf{q}}_r - \mathbf{C}\dot{\mathbf{q}} - \mathbf{G}\right) + \dot{\hat{\mathbf{a}}}^T\mathbf{\Gamma}^{-1}\tilde{\mathbf{a}} \tag{4.47}$$

Now, suppose we introduce a control law of the form

$$\boldsymbol{\tau} = \mathbf{Y}\hat{\mathbf{a}} - \mathbf{K}_D\mathbf{s} \tag{4.48}$$

where $\hat{\mathbf{a}}$ is the estimate of the parameter vector, and the dynamic regressor is redefined slightly such that

$$\mathbf{Y}\hat{\mathbf{a}} = \hat{\mathbf{M}}\ddot{\mathbf{q}}_r + \hat{\mathbf{C}}\dot{\mathbf{q}}_r + \hat{\mathbf{G}}. \tag{4.49}$$

Substituting this control into Equation (4.47) gives an expression for $\dot{V}$ as

$$\dot{V} = \mathbf{s}^T\mathbf{Y}\tilde{\mathbf{a}} - \mathbf{s}^T\mathbf{K}_D\mathbf{s} + \dot{\hat{\mathbf{a}}}^T\boldsymbol{\Gamma}^{-1}\tilde{\mathbf{a}}. \tag{4.50}$$

Defining an adaptation law for the parameters as

$$\dot{\hat{\mathbf{a}}} = -\boldsymbol{\Gamma}\mathbf{Y}^T\mathbf{s} \tag{4.51}$$

with a positive definite adaptation gain matrix,$\boldsymbol{\Gamma} \in \mathbb{R}^{p\times p}$, the last term of (4.50) cancels with the first term, leaving

$$\dot{V} = -\mathbf{s}^T\mathbf{K}_D\mathbf{s} \leq 0. \tag{4.52}$$

The remainder of the proof, detailed in [26] and shown below for the flexible controller, invokes Barbalat's lemma to show that $\mathbf{s} \to \mathbf{0}$ as $t \to \infty$, which implies, through the structure of the sliding variable, that the tracking error $\tilde{\mathbf{q}}$ and $\dot{\tilde{\mathbf{q}}}$ go to $\mathbf{0}$ as well.

An important aspect of the stability argument is the availability of a fully actuated control input, $\boldsymbol{\tau}$. Without full actuation, the control input can only implement part of the feedforward acceleration provided by $\mathbf{Y}\hat{\mathbf{a}}$, and therefore the simplification from Equation (4.47) to (4.50) is no longer possible. Therefore, in its current form, the control law is not suited for underactuated systems. The next section presents an alternative form that helps to address this problem.

4.4.2 Adaptive Control for Flexible Beam Maneuvering

Recalling the lessons learned from the design of the PFL controller, a fully actuated adaptive controller does not work for an underactuated system because it attempts to achieve perfect tracking for all states simultaneously. From this perspective, an adaptive method for the robot-beam system should incorporate the idea of a lower dimensional task space. One method for incorporating a task space representation called normal form augmentation is proposed by Gu et al. for adaptive control of free-floating robotic manipulators in Cartesian space [9]. The idea is extended here

to accommodate the task space defined in Equation (4.34) for the PFL controller. To simplify the derivation and develop a controller with a compact representation, the simplified beam model is used for the dynamics.

Starting with a desired output equation $\mathbf{y} = \mathbf{h}(\mathbf{q}) \in \mathbb{R}^m$, where m is the dimension of the control input, we augment the output with the unactuated coordinates

$$\mathbf{y}_a = \begin{pmatrix} \mathbf{y} \\ \boldsymbol{\delta} \end{pmatrix} \tag{4.53}$$

Now $\mathbf{y}_a \in \mathbb{R}^n$ has the same dimension as the coordinate vector $\mathbf{q}$. Differentiating Equation (4.53), we obtain an expressions for the velocity and acceleration of the augmented output in terms of the robot coordinates

$$\dot{\mathbf{y}}_a = \begin{bmatrix} \mathbf{J}_x & \mathbf{J}_\delta \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{pmatrix} \dot{\mathbf{q}} \\ \dot{\boldsymbol{\delta}} \end{pmatrix} = \mathbf{J}_{sq}\dot{\mathbf{q}} \tag{4.54}$$

$$\dot{\mathbf{y}}_a = \mathbf{J}_{sq}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{sq}\dot{\mathbf{q}} \tag{4.55}$$

A new Jacobian, $\mathbf{J}_{sq}$, transforms robot coordinate velocities to augmented output velocities. For the remainder of the derivation, it is assumed that the output function is the same as Equation (4.34), and the simplified beam model is used to represent the dynamics. The augmented output vector and Jacobian for this combination is

$$\mathbf{y}_a = \begin{bmatrix} x & y & (\theta_b + \alpha\delta_f) & \delta_f \end{bmatrix}^T \tag{4.56}$$

$$\mathbf{J}_{sq} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.57}$$

and $\dot{\mathbf{J}}_{sq} = \mathbf{0}$. The Jacobian is full rank, so the relationship between robot coordinates and augmented output can be inverted to give

$$\dot{\mathbf{q}} = \mathbf{J}_{sq}^{-1}\dot{\mathbf{y}}_a \quad \text{and} \quad \ddot{\mathbf{q}} = \mathbf{J}_{sq}^{-1}\ddot{\mathbf{y}}_a. \tag{4.58}$$

Defining $\bar{\mathbf{M}} = \mathbf{J}_{sq}^{-T}\mathbf{M}\mathbf{J}_{sq}^{-1}$, and $\bar{\mathbf{C}} = \mathbf{J}_{sq}^{-T}\mathbf{C}\mathbf{J}_{sq}^{-1}$, the robot dynamics are rewritten in the augmented output coordinates as[1]

$$\bar{\mathbf{M}}\ddot{\mathbf{y}}_a + \bar{\mathbf{C}}\dot{\mathbf{y}}_a + \mathbf{K}\mathbf{y}_a = \mathbf{J}_{sq}^{-T}\boldsymbol{\tau}. \tag{4.59}$$

To develop an adaptive controller from these dynamics, we start by defining the trajectory. The desired trajectory for the augmented output is modified slightly to produce

$$(\mathbf{y}_a)_d = \begin{pmatrix} \mathbf{y}_d \\ \delta_f \end{pmatrix}.$$

Instead of explicitly defining a trajectory for δ_f as we would have in the fully actuated case, the desired trajectory in this case is the current measured value. In this way we let the dynamics "compute" the appropriate value of δ_f. from the current trajectory. The sliding variable, $\mathbf{s}$, is defined, as before, as a filtered tracking error

$$\mathbf{s} \;=\; \dot{\tilde{\mathbf{y}}}_a + \boldsymbol{\Lambda}\tilde{\mathbf{y}}_a = \begin{pmatrix} \dot{\tilde{\mathbf{y}}} - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \\ 0 \end{pmatrix} \tag{4.60}$$

The velocity error representation, $\mathbf{s} = \dot{\mathbf{y}}_a - (\dot{\mathbf{y}}_a)_r$, results in

$$(\dot{\mathbf{y}}_a)_r = \begin{pmatrix} \dot{\mathbf{y}}_d - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \\ \dot{\delta}_f \end{pmatrix} \quad \text{and} \quad (\ddot{\mathbf{y}}_a)_r = \begin{pmatrix} \ddot{\mathbf{y}}_d - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \\ \ddot{\delta}_f \end{pmatrix} \tag{4.61}$$

The controller derivation can now proceed as in Equations (4.44) through (4.52). Starting with the Lyapunov candidate

$$V = \frac{1}{2}\left[\mathbf{s}^T\bar{\mathbf{M}}\mathbf{s} + \tilde{\mathbf{a}}^T\boldsymbol{\Gamma}^{-1}\tilde{\mathbf{a}}\right] \tag{4.62}$$

[1] $\mathbf{K}$ is not modified because it only multiplies δ_f, which remains in the same position in the augmented output vector

and differentiating with respect to time, we have

$$\dot{V} = \mathbf{s}^T \left(\mathbf{J}_{sq}^{-T} \begin{pmatrix} \boldsymbol{\tau} \\ \mathbf{0} \end{pmatrix} - \bar{\mathbf{M}}(\ddot{\mathbf{y}}_a)_r - \bar{\mathbf{C}}(\dot{\mathbf{y}}_a)_r - \mathbf{K}\mathbf{y}_a \right) + \dot{\hat{\mathbf{a}}}^T \boldsymbol{\Gamma}^{-1} \tilde{\mathbf{a}}. \tag{4.63}$$

The control law from [9] is defined as

$$\mathbf{J}_{sq}^{-T} \begin{pmatrix} \boldsymbol{\tau} \\ \mathbf{0} \end{pmatrix} = \mathbf{Y}\hat{\mathbf{a}} - \begin{pmatrix} \mathbf{K}_D \left(\dot{\tilde{\mathbf{y}}} - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \right) \\ \xi \end{pmatrix} \tag{4.64}$$

with the dynamic regressor

$$\mathbf{Y}\left(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{y}_a, (\dot{\mathbf{y}}_a)_r, (\ddot{\mathbf{y}}_a)_r\right) \hat{\mathbf{a}} = \hat{\bar{\mathbf{M}}}(\ddot{\mathbf{y}}_a)_r + \hat{\bar{\mathbf{C}}}(\dot{\mathbf{y}}_a)_r + \hat{\mathbf{K}}\mathbf{y}_a$$

The term ξ is added for notational convenience to allow the bottom part of

$$\begin{pmatrix} \boldsymbol{\tau} \\ \mathbf{0} \end{pmatrix} = \mathbf{J}_{sq}^{-1} \left[\mathbf{Y}\hat{\mathbf{a}} - \begin{pmatrix} \mathbf{K}_D \left(\dot{\tilde{\mathbf{y}}} - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \right) \\ \xi \end{pmatrix} \right]$$

to evaluate to $\mathbf{0}$ and does not enter into the control law. Substituting into Equation (4.63),

$$\dot{V} = \mathbf{s}^T \mathbf{Y}\tilde{\mathbf{a}} - \mathbf{s}^T \begin{pmatrix} \mathbf{K}_D \left(\dot{\tilde{\mathbf{y}}} - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \right) \\ \xi \end{pmatrix} + \dot{\hat{\mathbf{a}}}^T \boldsymbol{\Gamma}^{-1} \tilde{\mathbf{a}} \tag{4.65}$$

Applying the same adaptation law as Equation (4.51), the result is

$$\dot{V} = -\mathbf{s}^T \begin{pmatrix} \mathbf{K}_D \left(\dot{\tilde{\mathbf{y}}} - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \right) \\ \xi \end{pmatrix} = - \left(\dot{\tilde{\mathbf{y}}} - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \right) \mathbf{K}_D \left(\dot{\tilde{\mathbf{y}}} - \boldsymbol{\Lambda}\tilde{\mathbf{y}} \right) \leq \mathbf{0} \tag{4.66}$$

To show that the output tracking tends to zero, we must invoke Barbalat's lemma. Following the approach in [26], we differentiate again to obtain

$$\ddot{V} = -2\mathbf{s}^T \mathbf{K}_D \dot{\mathbf{s}} \tag{4.67}$$

Since V is positive everywhere except $\mathbf{s} = \mathbf{0}$ and $\tilde{\mathbf{a}} = \mathbf{0}$, if $\ddot{V}$ is shown to be bounded, then

$$\ddot{V} \text{ bounded } \Rightarrow \dot{V} \rightarrow 0 \Rightarrow \mathbf{s} \rightarrow \mathbf{0}$$

With $\dot{V} \leq 0$ and $V \geq 0$, V is bounded, which implies that both $\mathbf{s}$ and $\tilde{\mathbf{a}}$ are bounded from the definition of V in Equation (4.62), so $\ddot{V}$ is bounded as well. Therefore, we can conclude that the output tracking error converges to $\mathbf{0}$. For global stability of the dynamics, we must add the additional condition that the zero dynamics related to the output equation are stable, just as was required for the PFL controller.

As a consequence of adding the measured value of δ_f to the desired trajectory, the reference acceleration, $(\ddot{\mathbf{y}}_a)_r$, now requires a value for $\ddot{\delta}_f$. There are several approaches to obtaining this measurement:

Differentiation By augmenting the beam angle estimator from Section 3.4 with another layer of differentiation, the deflection angle acceleration can be computed from the beam measurements of δ_f.

Calculation Using an assumed model of the dynamics, the partial feedback linearization from Equation (4.37) can be used to compute the deflection angle acceleration from the measured values of δ_f, $\dot{\delta}_f$, and the output reference acceleration, $\ddot{\mathbf{y}}_r$.

Elimination Considering the acceleration contribution to be small, we may neglect it completely.

The first technique is subject to high noise levels due to the repeated differentiation. In simulation, eliminating the acceleration does not appear to adversely affect the tracking performance, and this approach is used for implementations of the control law in this study. A proof of stability with the acceleration removed remains as future work.

The final component of the adaptive control law is an update method for tuning the weighting parameter α online. In [28], Talebi et al. use a gradient learning method

based on the beam deflection. Defining the output error

$$e = y_{beam} - (y_{beam})_r \tag{4.68}$$

and a cost function

$$E = \frac{1}{2}e^2 \tag{4.69}$$

we can implement a learning rule of the form

$$\dot{\alpha} = -\eta \frac{\partial E}{\partial \alpha} \tag{4.70}$$

where η is a type of gain called the learning rate. Taking the partial derivative results in

$$\dot{\alpha} = -\eta e \delta_f. \tag{4.71}$$

This update modifies α whenever there is output tracking error and there is a nonzero beam deflection. In simulation results, adding this adaptation improved the damping performance of the adaptive controller.

4.4.3 Adaptive Control with the Simplified Beam Model

An important benefit of using the simplified model as a template comes from the way rigid bodies are represented in the adaptive control law. In 2D, a single rigid body has four physical parameters: mass, inertia, and the two center of mass coordinates relative to the origin of the body. The simplified beam model consists of two rigid bodies connected by a spring with an unknown spring constant, for a total of 9 parameters, each of which are adapted in the control law to achieve a tracking goal. For the first body (the base of the robot), we must know *a priori* the point where the second body (the beam) connects to determine the effect of forces caused by the beam's motion. This is not a significant restriction because we assume that the physical layout of the assembly vehicle is known beforehand and that the beam attaches to a pre-defined location. More importantly, we do not need any additional information about the

second body because the parameters responsible for its dynamic interaction with the first body are all part of the adaptive control law. In industrial robotics the same property is used to adapt for unknown payloads carried by manipulators, which are essentially extensions of the last link of the robot arm.

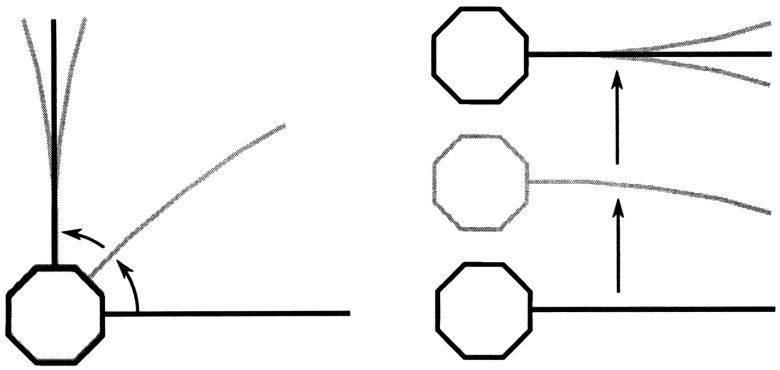
Provided the simplified dynamic model continues to be a good approximation to the selected structural element and a trajectory is chosen that satisfies actuator limitations, the adaptive control system should be capable of stably manipulating beams of a variety of lengths without modification. In the same way, the controller is also capable of maneuvering an arbitrary rigid structure attached to the robot base.

4.5 Simulation Results

4.5.1 Baseline Trajectories

As described in Section 1.2, there are three phases in the docking process: approach, berthing, and capture. The first two phases are most significant for beam maneuvering because they require trajectory tracking and stabilization. Trajectories for these maneuvers are typically generated in a high level path planner, and clever path planning can result in significant vibration reduction. The precise shape of optimal maneuvers is beyond the scope of this study, but all maneuvers are fundamentally constructed from a combination of translations and rotations of the flexible element. The performance of each controller is evaluated in terms of its ability to track these two basic maneuvers.

To simulate a large angle reorientation of the beam, the rotation maneuver rotates the flexible element by 60 degrees (1.05 rad). In translation, we may choose any direction, but the best test is movement (0.75 m) in a direction perpendicular to the beam axis where beam oscillations are most strongly excited by a linear motion. The baseline trajectories are diagrammed in 4.2, and Figure 4.3 displays them as a function of time. Both trajectories follow a bang-off-bang profile, where an initial acceleration period initiates the movement, followed by a coasting section, and ending

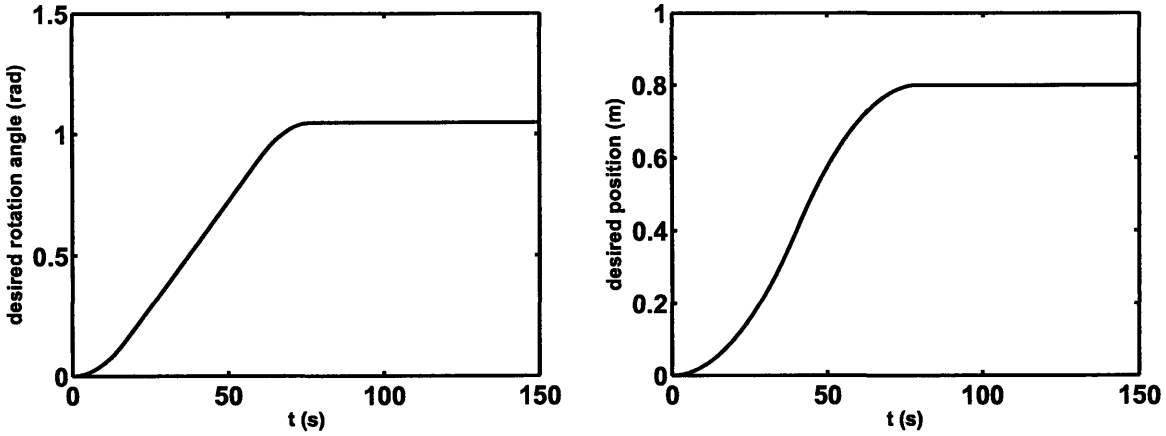


(a) (b)

Figure 4.2: The two baseline trajectories: (a) A rotation maneuver in which the satellite rotates by 60 deg. (b) A 0.5 m translation perpendicular to the beam to excite beam dynamics

with a deceleration at the desired target. Bang-off-bang profiles are an optimal path for a rigid body with a mixed time and fuel constraint, and they are used here to show the performance of the controllers both during transit and at the end of the maneuver.

The simulation models for both profiles are initialized with a small initial tracking error (5 degrees (0.1 rad) for the rotation maneuver and 0.1 meters for the translation maneuver) to show convergence to the desired trajectory. Rotation plots display the angle of the robot base, θ_b and not the augmented task space, $y_{beam} = \theta_b + \alpha\delta_f$. This allows us to see if the tracking controllers complete their trajectories at the desired rotation angle. In the following figures, the trajectory tracking performance is displayed along with the internal beam deflections as an indication of the damping performance. δ_1 is shown with dashes, δ_2 is shown with a solid line, and δ_3 is a dash-dot line. Finally, in this section, only rotation plots will be displayed to highlight important aspects of the controller performance. The remaining translation plots are located in Appendix A.



(a) (b)

Figure 4.3: Baseline Trajectories vs. Time

PD		PFL		Adaptive	
$\mathbf{K}_p$	$diag(0.58, 0.58, 0.32)$	$\mathbf{K}_p$	$diag(0.64, 0.64, 0.04)$	$\mathbf{K}_D$	$diag(2, 2, 2)$
$\mathbf{K}_d$	$diag(7.7, 7.7, 4.28)$	$\mathbf{K}_d$	$diag(2.4, 2.4, 0.6)$	$\mathbf{\Lambda}$	$diag(0.35, 0.350.1)$
		α	0.5	$\mathbf{\Gamma}$	$1000 \cdot \mathbf{I}_{9x9}$

Table 4.5.1: Control Parameters for Nominal Test

4.5.2 Nominal Model Tracking Performance

To compare the fundamental performance of the controllers, this set of simulations uses a known dynamic model with perfect state information. The PFL controller is constructed from the true nonlinear model, and the adaptive controller is initialized using the best guess for the simplified model approximations for inertia and spring constants from Chapter 2. A set of PD gains have been selected to achieve critical damping and a closed loop bandwidth of approximately 0.15 rad/s. To model the thrusters with limited disruptions to the ideal dynamics, applied forces and torques have been saturated at 0.025 N and 0.025 $N \cdot m$ and sampled at 0.4 Hz. A pulsed thruster model is used in the remaining sections. The key parameters for each controller are displayed in Table 4.5.1.

From the rotation trajectories displayed in Figure 4.4, we can see that the two tracking controllers provide good path following performance, but the adaptive controller has significantly better damping at the end of the trajectory. Though damping

was added to the simulation, the PFL controller is still quite oscillatory. To see why, we can examine the trajectory tracking performance in the task space, as shown in Figure 4.5. In the task space, the controller has nearly perfect tracking, which results in the system relying on the zero dynamics to decay slowly away after the target is reached. The adaptive controller performs much better overall even though it has a similar tracking goal. One explanation is the longer time to convergence in the task space. During this time, the controller adjusts its parameters to better attenuate vibrations or focuses more heavily on reducing the base rotation rate.

From these profiles, we already see that the adaptive control method is a promising control solution, possessing both tracking and vibration damping characteristics, but the performance must be viewed in the context of the idealized model. In particular, with no noise and a simple thruster model, we can use relatively high gains to achieve better tracking and damping performance. The next sections examines a more realistic case.

4.5.3 Performance with Estimated States

The simulation model presented in this section incorporates several details to improve accuracy. Thrusters are modeled with the pulse modulation described in Chapter 2, and the estimators from Chapter 3 are included to supply state information to the controllers. The PFL controller uses the full state Kalman filter from Section 3.3, and the adaptive controller uses the simplified deflection angle estimator from Section 3.4. For all controllers, including the PD controller, the global state measurements, (x_b, y_b, θ_b), and their derivatives are corrupted with white noise to simulated the global estimation system.

To achieve stability, the PFL controller uses a different gain set, listed in Table 4.5.2. The adaptive and PD controllers did not require an adjustment. These gains are frozen for the next simulation where the controllers are compared against an uncertain model.

Both the Adaptive controller and the PD controller achieve similar performance to the previous section. We expect this behavior because the PD controller should not

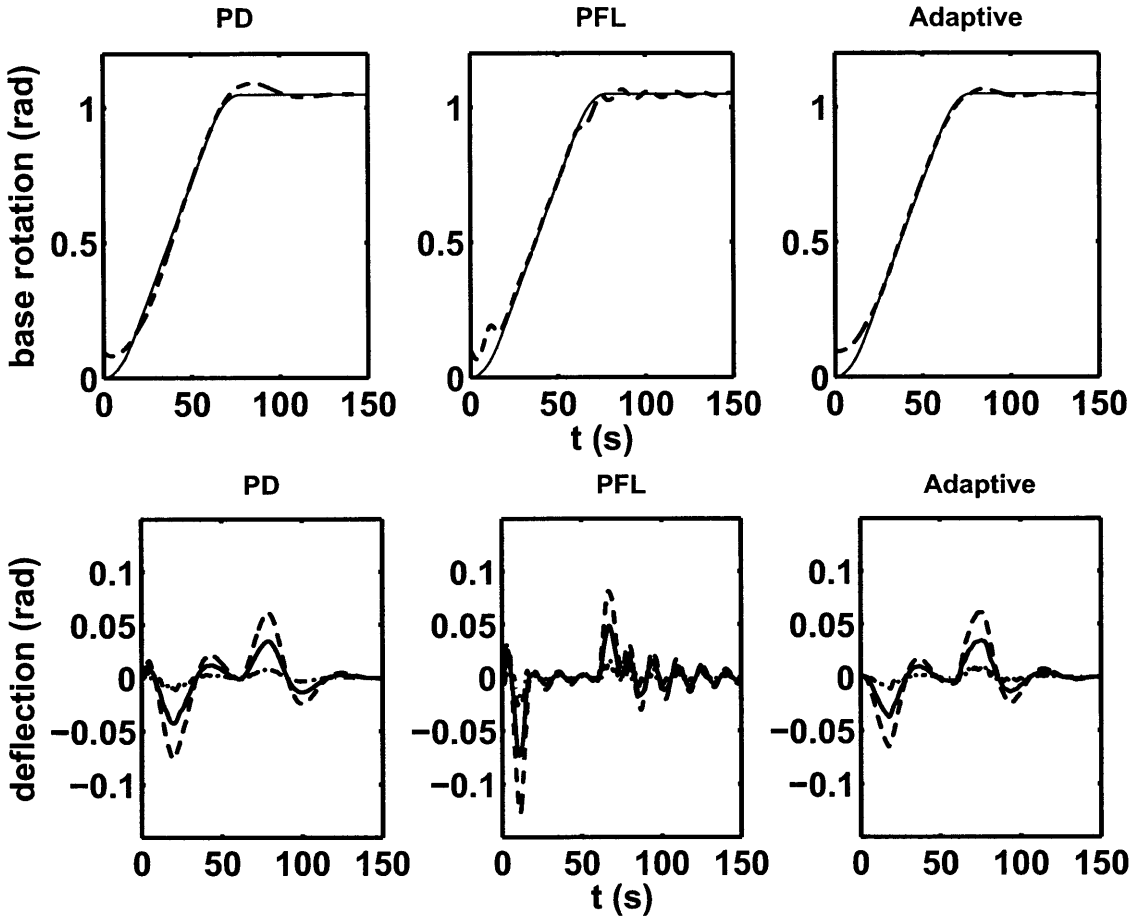


Figure 4.4: Both tracking controllers follow the rotation trajectories well with a nominal model, but the adaptive controller provides the best damping.

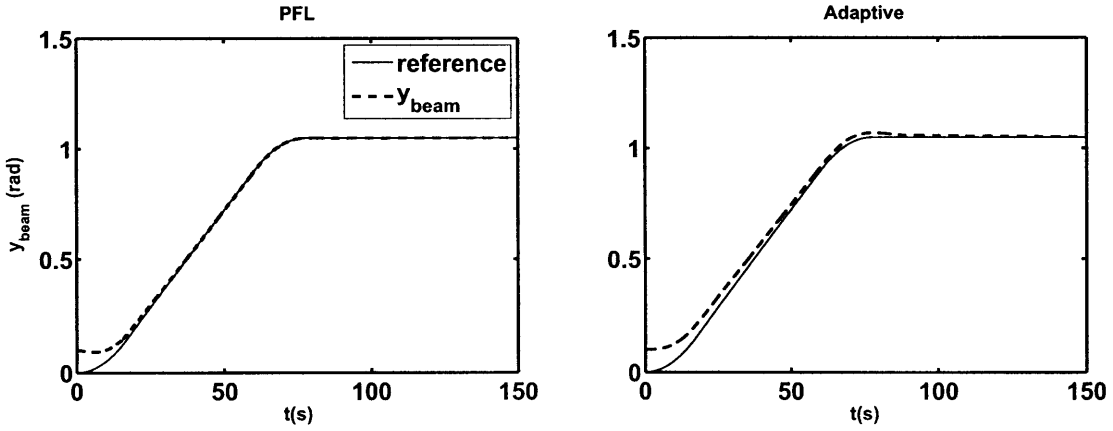


Figure 4.5: The PFL controller has excellent tracking in the task space, converging to the trajectory within 50 seconds, while the adaptive controller does not converge until about 125 seconds.

PD		PFL		Adaptive	
$\mathbf{K}_p$	$diag(0.58, 0.58, 0.32)$	$\mathbf{K}_p$	$diag(0.04, 0.04, 0.04)$	$\mathbf{K}_D$	$diag(2, 2, 2)$
$\mathbf{K}_d$	$diag(7.7, 7.7, 4.28)$	$\mathbf{K}_d$	$diag(0.6, 0.6, 0.6)$	$\mathbf{\Lambda}$	$diag(0.35, 0.350.1)$
		α	0.3	$\mathbf{\Gamma}$	$1000 \cdot \mathbf{I}_{9x9}$

Table 4.5.2: Control Parameters for Model Uncertainty Test

be significantly affected by the global estimation noise or the thruster approximation, and the adaptive controller can adjust to the mismatch between expected forces and actuated thrust.

The PFL controller shows much more oscillatory behavior, to which any of the following potential sources could contribute. Firstly, connecting the estimator to the PFL control signal tended to result in instabilities of the state estimate. The issue was avoided by disconnecting the control signal and using observations only to estimate the state. Since the PFL controller relies on a full state estimate, the approximate knowledge of the state may affect performance. Second, the PFL model has no way of compensating for the model uncertainties introduced by the thruster pulses. For stability, we end up relying on robustness characteristics of the controller that are not guaranteed to exist. Finally, the gain tuning performed for the PFL controller was aimed at producing a stable result, not an optimal solution. This approach is in line with the assumption that the objects being manipulated during assembly are uncertain and rigorous gain tuning prior to operations would not be available. We can conclude, as we have suspected before, that partial feedback linearization without additional augmentations is primarily useful when the model is well known and accurately estimated.

4.5.4 Performance with an Uncertain Model

Using the gains and and trajectories shown in the previous sections, the controllers are presented with a model containing large uncertainty in the total length of the beam. Each of the three outboard links of the beam was extend by 1.8 times its length with the remaining properties of the system held constant. Figure 4.7 shows the tracking performance, where we see that the PD controller is no longer properly tuned well

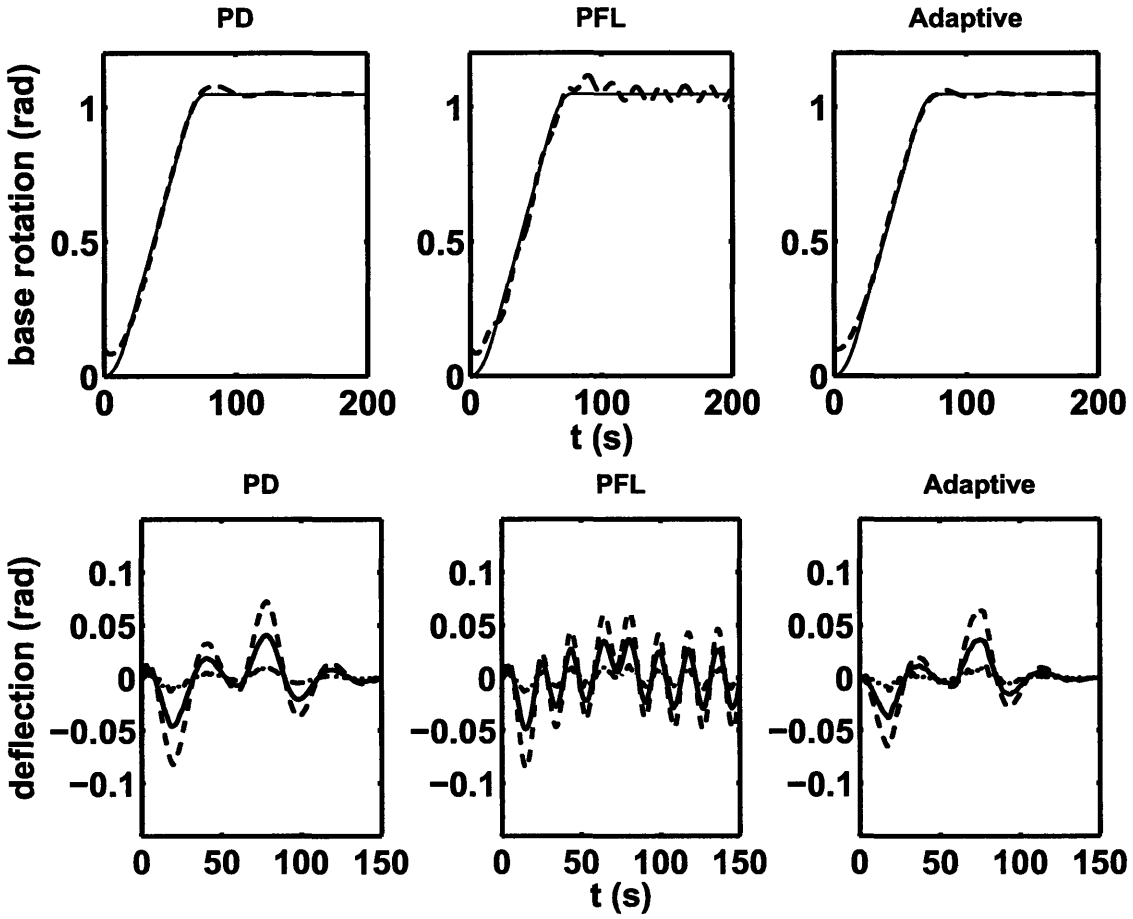


Figure 4.6: This figure shows tracking performance for a rotation profile when states are provided by a state estimator. PD uses global information, PFL uses the full state Kalman filter, and the Adaptive controller uses the beam deflection estimator.

for the dynamics. Though the properly tuned PD controller has compared favorably with the other controllers, we see that it is also susceptible to model changes.

The PFL controller appears to fare somewhat better in this test, likely because of the slower dynamics. An interesting point for future research is that the embedded model for the full state estimator in this scenario remained unchanged. This appears to have actually improved the performance of the PFL controller slightly, which warrants a closer look. Nonetheless, the PFL controller still suffers from significant overshoot compared to the adaptive controller, which manages to settle to the target position within about 50 seconds. The PFL profile also has a small, though sustained, oscillation in the beam angles present at the end of maneuver. As with the previous test, this test shows unreliable performance of the PFL controller under model uncertainty.

4.6 Hardware Results

Limited hardware testing was performed with a simpler version of the Adaptive control law to verify its damping performance in comparison to a PD controller. The controller is of the form

$$\boldsymbol{\tau}_x = \mathbf{Y}_x \hat{\mathbf{a}} - \mathbf{K}_D \mathbf{s}_x - \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} K_\delta \delta_f \tag{4.72}$$

where $\mathbf{s}_x$ is the sliding variable for the actuated states, and $\mathbf{Y}_x$ is the regressor for the actuated dynamics. This form is used for adaptive control of robots with flexible joints such as in [8]. Figure 4.8 compares an open loop damping decay profile to active control by a PD controller and the adaptive controller. Both controllers show active control of the beam vibrations, and the adaptive controller shows a faster response. This test gives promise for future implementation of the tracking controller presented above.

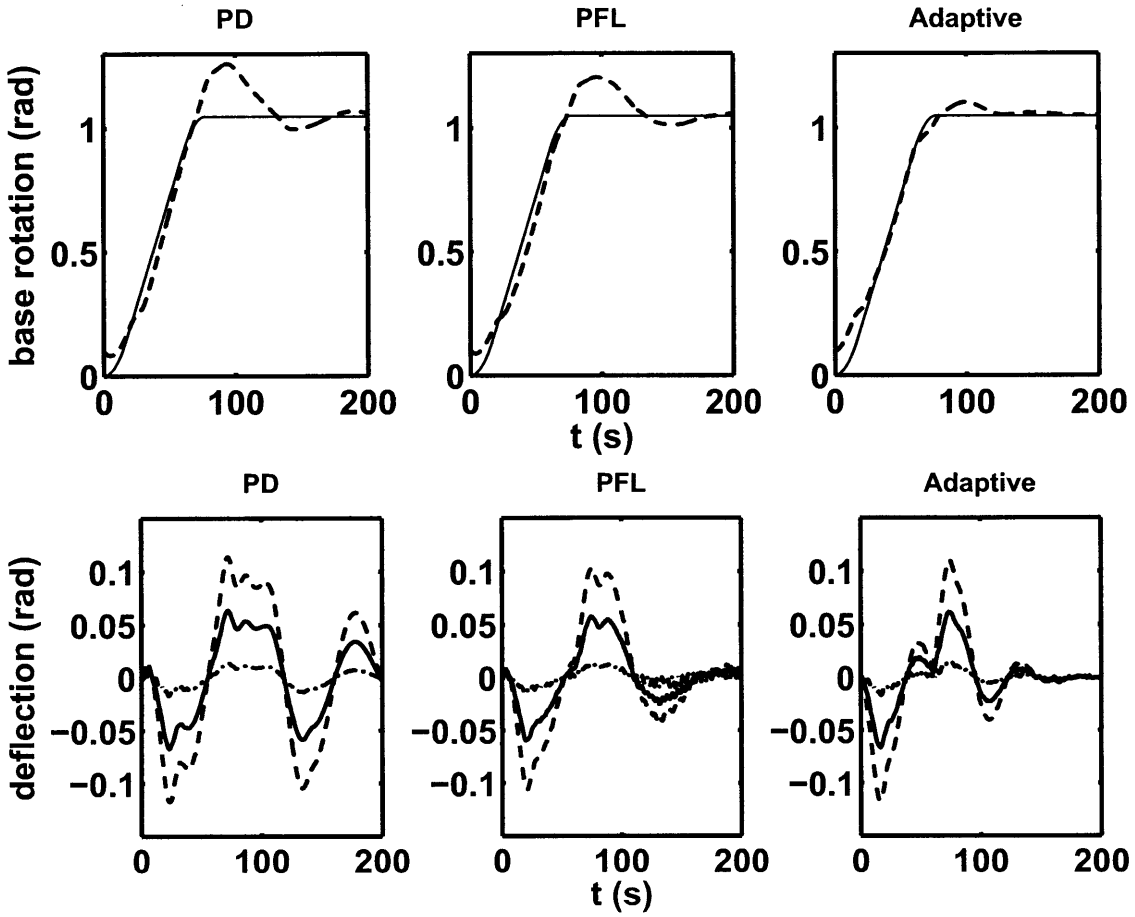


Figure 4.7: Tracking performance with a beam 1.8 times the length of the nominal beam. The PD controller is no longer well tuned for these dynamics.

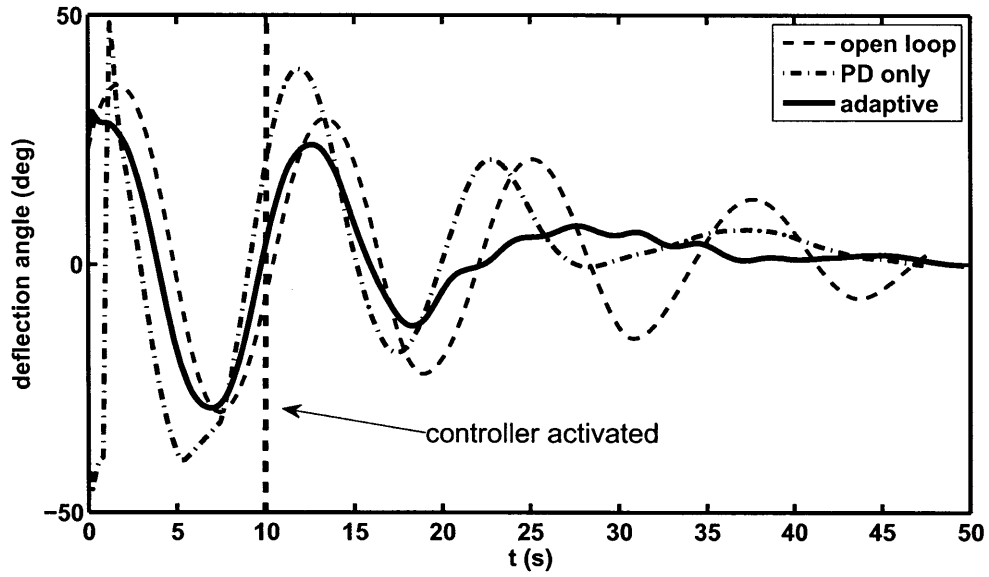


Figure 4.8: Adaptive control and PD control in hardware testing show active damping of the beam vibrations compared to an open loop decay profile.

4.7 Conclusions

This chapter presented three control systems to manipulate components in the assembly of flexible space structures. Each controller was evaluated for tracking performance, vibration attenuation, and robustness to uncertainty. A properly tuned PD controller performs well against more complicated model-based controllers but suffers in performance when used for plants that vary significantly from the initial implementation.

The partial feedback linearization controller provided a useful insight in the proper task space for an underactuated tracking controller. In terms of performance, PFL achieves excellent tracking for a known model, but it has marginal vibration attenuation and has no explicit way to compensate for model uncertainty. This observation is in line with other studies as PFL controllers are often used to achieve good tracking performance and not usually implemented for stabilization. Shkolnik implements a switching logic that engages a linear controller after the system has been navigated into the region of attraction [23]. This approach was the original path for implementing a PFL controller on the SWARM robot, but the adaptive control technique

proved to provide better performance and more versatility.

Adaptive control shows the best performance in simulation testing and initial hardware testing with good trajectory tracking, damping performance on par with or better than the PD controller, and most importantly, the ability to compensate for model uncertainty.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis presented the components of an estimation and control system for the assembly of flexible elements of a large space structure. In each chapter several approaches were examined to create a method capable of manipulating a variety of flexible payloads.

To develop an understanding of the behavior of flexible dynamic systems and provide a template for the control system design, Chapter 2 presented a dynamic model of a segmented beam using an Euler-Lagrange analysis. The nonlinear equations of motion were arranged in a form that shares similarities with homogeneous flexible beams and robot manipulators. From the full nonlinear dynamics, a linearized model and a single joint flexible beam model were introduced as simplifications. The simplified model proved to show good performance as a template for estimation and control.

Chapter 3 introduced two estimation systems, based on the simplified dynamic models, for observing the dynamics of a flexible beam using vision measurements. A linear Kalman filter approach was used to extract estimates of the internal beam joint angles, while a steady-state filter was used to estimate the angular position and angular velocity of the total beam deflection. With the Kalman filter it was determined that the full state of a vibrating beam can be extracted from observations

of the tip movement and gyro measurements, but adding model uncertainty can disrupt the state estimates. Without making model updates to the estimator, it is difficult to maintain consistent performance. The best performance was obtained with the beam deflection filter, which measures the beam deflection directly and provides of filtered differentiation of the beam deflection angle.

Chapter 4 examined three approaches for using measurement and model information to compute a control signal. PD control, Partial Feedback Linearization, and Adaptive control were compared for their performance in trajectory tracking and vibration suppression. The partial feedback linearization approach for control shares some of the same model uncertainty issues as the beam estimator. Though the PFL controller shows excellent tracking of a selected task space trajectory with a known model and good measurements, it requires full state information and a reasonably accurate model to control the beam. Applying the control law with estimated states from the linear Kalman filter and pulsed thrusters resulted in unpredictable performance. Therefore, unless there is strong confidence in the dynamic model, the PFL control law should not be used. Although the PFL approach is not well suited for implementation, examining the choice of task space provides an interesting insight into vibration control. Choosing a combination of base rotation angle and weighted deflection angle results in a measurement with stable zero dynamics.

In contrast to PFL, the adaptive control law paired with the deflection angle estimator shows consistent performance in scenarios incorporating model uncertainty and realistic measurements. These characteristics match well with the requirements for on-orbit assembly, where the ability to adjust to system properties allows for minimal reconfiguration of the control system. Adaptive controllers show promise in initial hardware testing and will be used for future investigations.

5.2 Future Work

5.2.1 Extending to 3D

As a first attempt for demonstrating robotic assembly in a hardware demonstration, the estimation and control laws presented in this study were developed for a 2D planar robot. Extensions to three dimensions and 6DOF maneuvers presents many additional challenges, though they may still be addressed using the framework presented here.

The vision-based measurements of beam deflection covered in Chapter 3 can be extended to three dimensions by including a calculation of both the X' and Y' distances in Equations (3.2) and (3.3), shown below:

$$\begin{aligned} \frac{X'}{Z'} &= \beta\frac{\bar{x} - x_0}{f} \\ \frac{Y'}{Z'} &= \beta\frac{\bar{y} - y_0}{f} \end{aligned}$$

Whereas in Section 3.2 either the depth measurement, Z', or Y' had to be known, in this case, Z' must be known beforehand. Calculating both X' and Y' provides the 3D location of the beam tip in the camera frame.

The controls and dynamics for 3D motion are more complicated. Rotational motion becomes nonlinear, and the vibrational dynamics of the beam are coupled to more degrees of freedom. We may continue to use the useful analogies provided by industrial robotics to develop the full equations of motion for the 3D system, and the Euler-Lagrange derivation is still applicable. Several researchers (e.g. [9]) have considered the problem of moving a robot arm in space with an unactuated base, and this problem is similar except that the arm is unactuated. Slotine and Li proposed an adaptive algorithm for spacecraft attitude control in [26], and using the modeling approach in Chapter 2, it should be possible to extend this model with a flexible appendage. From there the adaptive control law can be modified with the normal form augmentation method shown in Chapter 4.

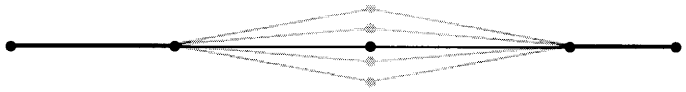


Figure 5.1: The vibration mode shown above does not cause a large deflection of the endpoint.

5.2.2 Estimator Improvements

Camera Corrections

It is possible to perform image calibrations that determine a camera's pixel size, focal length, and lens distortion. Obtaining more accurate values for the pixel size and focal length, and applying corrections to compensate for lens distortions would improve the quality of beam measurements.

Multiple Beam LEDs

Although tip deflection measurements provide observability of the beam state, the measurement accuracy and sensitivity to noise could be significantly improved by adding one or more LEDs to other locations on the beam. In particular, consider the diagram in Figure 5.1. The vibration of the middle segments of the beam does not cause a large deflection of the tip of the beam and therefore is not easily observable from the tip LED. Placing an LED in the center of the beam would add much more information about this vibration. For implementation with a true flexible beam, several LEDs may be necessary.

Multiple LEDs require minimal modification to the control and estimation schemes presented in Chapter 3. The image processing algorithm must be augmented with a method for dividing the field of view into several regions, each of which contain a single dot. After isolating the individual LEDs, the positions can be determined using the centroiding algorithm. For the Kalman filter measurement equations, a new row is added to the $\mathbf{C}$ matrix for each of the new LEDs reflecting the position as a linear function of the beam angles.

Adaptive Estimation

An important simplifying assumption in the development of the control system presented in this thesis was the accuracy of the simplified dynamic model in representing the main dynamic behavior of the flexible beam. If this model did not produce a sufficient representation of the beam dynamics, it would be necessary to use more detailed models of the dynamics along with controllers that are either model-free or capable of incorporating the detailed models. Unfortunately, using a detailed model destroys some of the versatility gained by using an adaptive controller, particularly if there are multiple structures to maneuver that require unique but detailed models. The problem extends to producing an accurate state estimate in the event that a full state measurement is required.

One solution for addressing this problem is to extend the idea of adaptive control to an adaptive estimation system. By parameterizing the embedded model in the estimator, it may be possible to use an external parameter identification scheme to make updates to the embedded model. This approach would likely require an initial excitation phase where the control system applies a set of persistently exciting open loop signals to determine the relevant dynamics. After identification, the model could be used to synthesize a controller online.

Nonlinear Observers

The estimators explored in this thesis are all based on linear models, but nonlinear estimators may be required for more aggressive maneuvers and full 6DOF motion. The Extendend Kalman Filter (EKF) would be a first step to update the linear filters presented in this study. The EKF uses update equations based on the linearization of the equations of motion about the current state estimate. Given the symbolic linearizations of the beam dynamics, it would be possible to construct a filter that uses both nonlinear dynamics and nonlinear measurements, thereby eliminating the small angle assumptions used in Section 3.2.

5.2.3 Controller Improvements

Discrete Time Compensation

The controllers implemented in Chapter 4 used continuous-time control laws sampled at discrete intervals. Though simulations showed this method was stable and achieved reasonable results, the stability proofs do not extend to sampled data systems. In [30], Warshaw proposed a modification of Slotine's adaptive control architecture along with a stability proof for sampled data systems. The modification adds an additional term to the control law to compensate for the discretization. Talebi et al. proposed a neural network architecture in [29]that operates directly in discrete time and thus avoid the need for a continuous time compensation. Both approaches should be considered for adding stability guarantees to the control law.

Composite Adaptation

Though adaptive control allows the manipulation of a variety of structures, the adaptation law presented in this thesis is driven only by trajectory tracking error and therefore only modifies the adaptive parameters until the tracking error is small. With this form of update, the parameters only converge to their true values in the case where the trajectory is "sufficiently rich" to require all parameters to be correct for perfect tracking. For many applications, this property is considered a feature because the parameter knowledge only needs to be good enough to follow the desired trajectory and does not require the energy for a full system identification.

A modified approach, called *composite adaptation* [26], adds another layer of adaptation by using prediction error as a second source of information. Instead of only relying on the adaptive model to calculate forces and torques from the tracking error, we may also use the model to predict forces and torques from measured motion:

$$\hat{\boldsymbol{\tau}} = \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\hat{\mathbf{a}}.$$

After filtering the control prediction to remove the need for joint acceleration, we can

compare with the the calculated control signal, using the error in a second update to drive the adaptation law. In this way we take advantage of both the trajectory tracking error and the model prediction error to improve the estimate of the parameters. Additional implementation details may be found in [26].

Robust Control

The exploration of model uncertainty in this study focused on errors caused by poorly known parameters. In addition to parameter uncertainty, errors in modeling of the dynamics may also contribute significantly to tracking error and vibration control. For example, the simplified beam model approximates a flexible beam with a single flexible joint, which does not exactly represent the true dynamics. Truncated models for an Euler-Bernoulli beam make similar errors by representing the infinite number of vibrational modes with a finite number.

Robust control attempts to address model uncertainty by placing a bound on the expected error and applying a compensator to account for the error. A popular approach for nonlinear systems is sliding mode control[26, 25], which has a control law of the form

$$\begin{aligned} \hat{\boldsymbol{\tau}} &= \hat{\mathbf{H}}\ddot{\mathbf{q}}_r + \hat{\mathbf{C}}\dot{\mathbf{q}}_r + \hat{\mathbf{G}} \\ \boldsymbol{\tau} &= \hat{\boldsymbol{\tau}} - \mathbf{k}sgn(\mathbf{s}) \end{aligned}$$

Where $\hat{\boldsymbol{\tau}}$ is an estimated control input based on an approximate model of the system, and $\mathbf{k}$ is a gain term such that

$$k_i \geq \left| \left[\tilde{\mathbf{H}}\ddot{\mathbf{q}}_r + \tilde{\mathbf{C}}\dot{\mathbf{q}}_r + \tilde{\mathbf{G}} \right]_i \right|.$$

A control law of this form can be shown to asymptotically drive the tracking error $\mathbf{s}$ to zero. Extensions can be derived for adaptive control, where the adaptive control law is modified with a deadband to prevent adaptation from noisy or uncertain data.

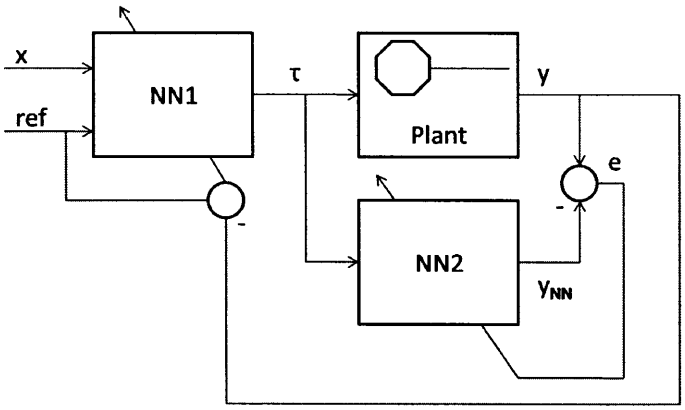


Figure 5.2: The error between the measured output of the plant and NN2 is used to train NN2, and the error from the reference signal is used to train NN1.

Neural Control

Though this study took a model-based approach, function approximation techniques such as neural network control may be a promising alternative for providing a versatile control system. Instead of using a model of the system to calculate control inputs, a neural approach would use a network of nonlinear functions

$$\tau_i = \sum_j v_{ij} f\left(\mathbf{x}, \mathbf{w}_j\right)$$

where v_{ij} are weights on the functions f, and $\mathbf{w}_i$ are weights on the inputs $\mathbf{x}$. The network weights are often trained either online or offline using a technique called backpropagation, which adjusts the weights in order to reduce an error function related to the output of the network. For instance, while controlling the flexible beam we may choose the squared beam deflection as an output of interest and train the network to reduce the error while manipulating the beam. This requires knowledge of the relationship between the output of the network (the control input) and the beam deflection determined by the plant dynamics. One approach is to use a second neural network to approximate the plant as shown in Figure 5.2. In [28], Talebi et al., present a variety of methods using neural networks for control of a single link flexible manipulator, and many of the techniques could be applied to the beam manipulation problem.

Appendix A

Additional Simulation and Hardware Testing

A.1 Simulated Translation Profiles

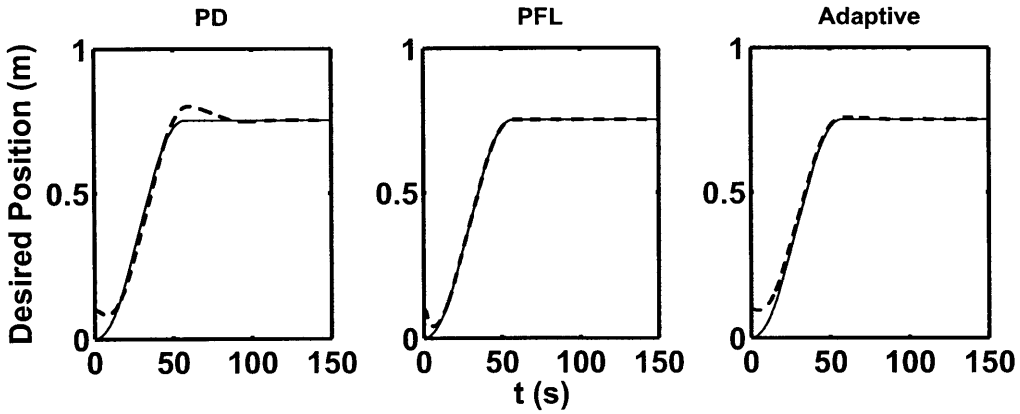


Figure A.1: Tracking Performance for Translation with Nominal Model

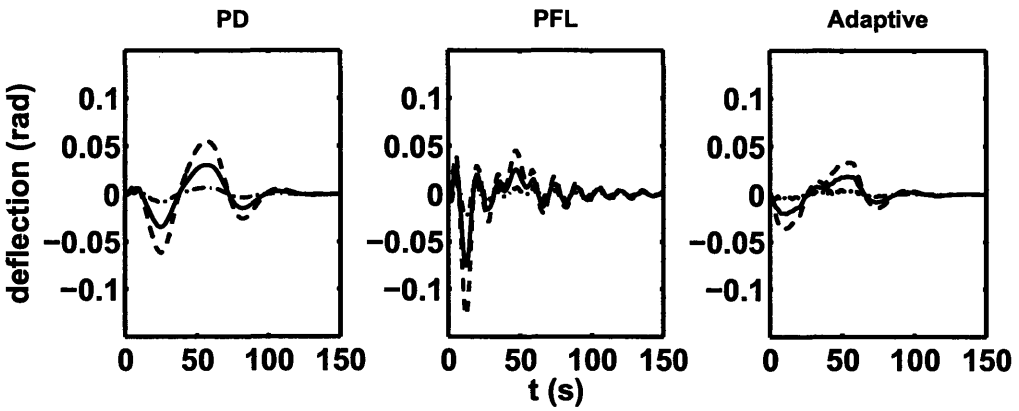


Figure A.2: Angle Deflections for Translation with Nominal Model

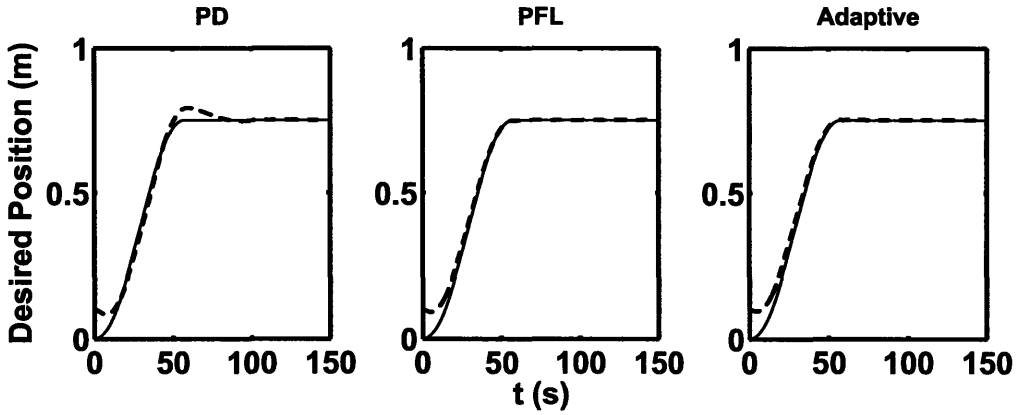


Figure A.3: Translation Profile with Estimated States

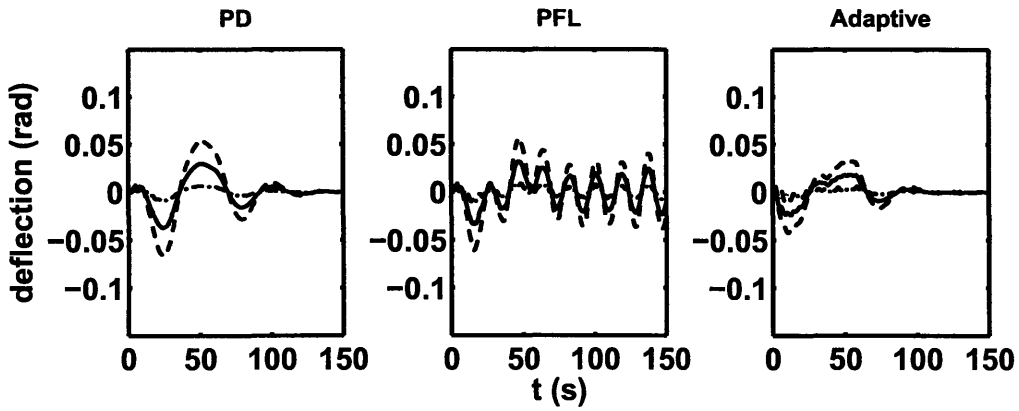


Figure A.4: Angle Deflections for Translation with Estimated States

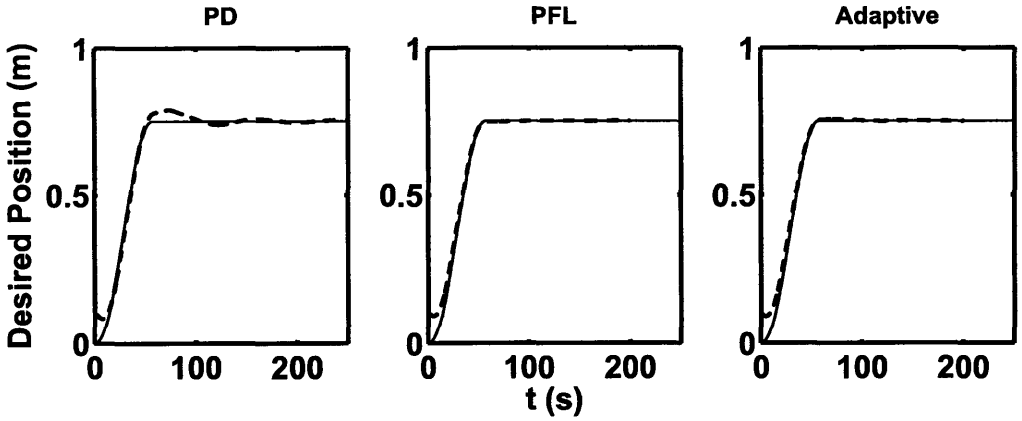


Figure A.5: Translation Profile with Long Beam

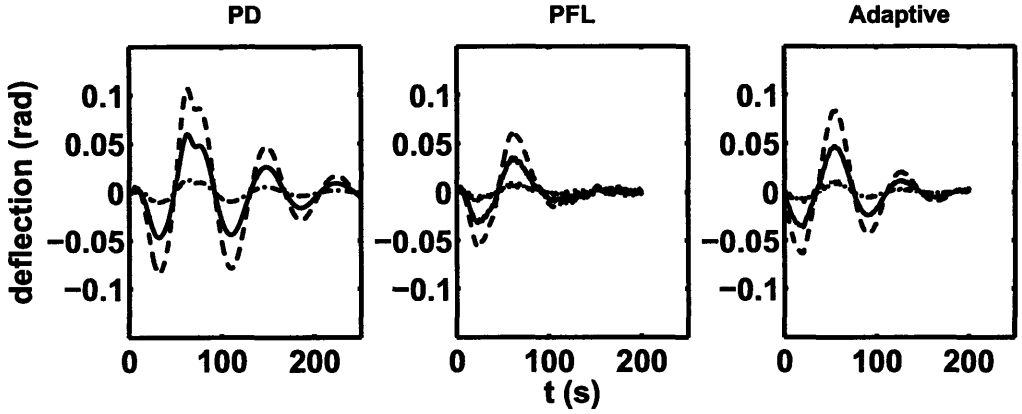


Figure A.6: Angle Deflections for Long Beam Translation

A.2 Hardware Testing

A.2.1 SWARM Assembly Scenarios

The SWARM Phase II test program consisted of several tests designed to demonstrate key aspects of the assembly process. As shown in Figure A.7, testing started with a demonstration of controlled maneuvering of the flexible element and culminated in a full assembly of a simple structure consisting of two satellites joined by a flexible beam. The tests are summarized below:

Test 1 demonstrated rotation and translation maneuvers for a beam docked to the propulsion module.

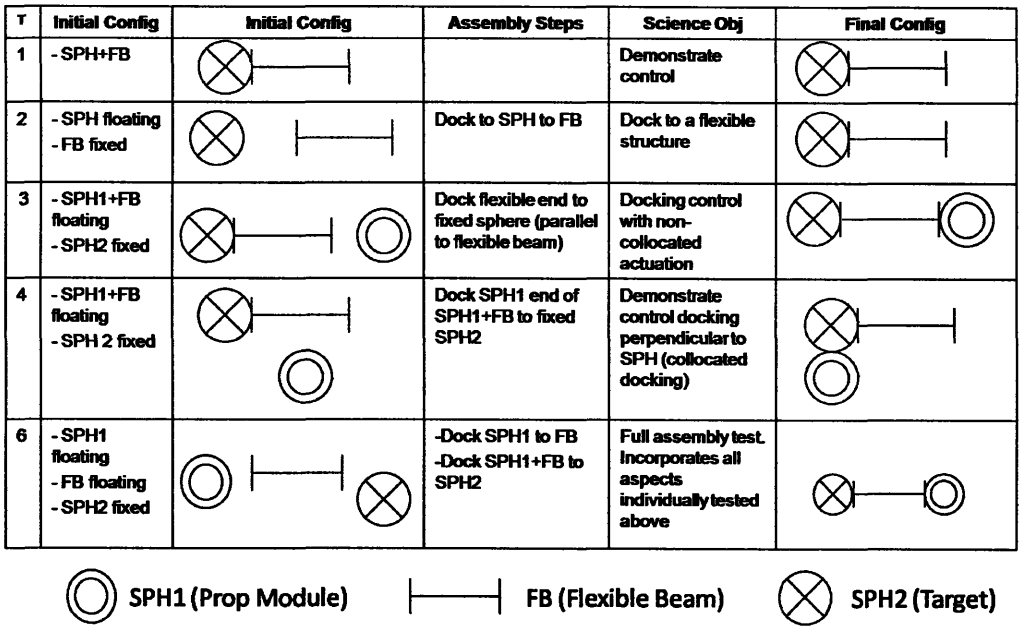


Figure A.7: A graphical summary of the SWARM Phase II test program.

Test 2 demonstrated the ability of the propulsion module to dock to a beam and engage the docking mechanism.

Test 3 demonstrated noncollocated docking of the beam end to a fixed target with the beam clamped to the propulsion module.

Test 4 demonstrated a collocated docking of the propulsion module to a fixed target.

Test 5 (not shown) demonstrated another docking with the beam docked to the propulsion module.

Test 6 demonstrated a complete assembly, starting with 3 separate pieces and finishing with an assembled structure.

A.2.2 Hardware Results

The plots below are results from hardware testing of the scenarios described above. Test 1 is not included because the beam maneuvering results are available in the other tests. The following guide can be used to interpret the results:

- Vertical * symbols or thick vertical bars indicate a change of maneuver such as switching from a approach to berthing or from a position hold to a translation maneuver

- Horizontal * symbols or thick solid bars indicate the location in the X direction or the Y direction where docking is expected to occur. Past this line is inside the target.

- Desired trajectories can be distinguished from actual trajectories by the noise and oscillation in the actual trajectories.

The path planner provided trajectories similar to the basic translation and rotation trajectories presented in Chapter 4 at the beginning of each maneuver. Paths were planned from the current position to the target position, which accounts for the jumps in the desired trajectory at the beginning of each maneuver.

All tests were performed with an early version of adaptive control law based on a method described by Shahravi in [22]. The adaptive algorithm did not perform as well as expected, and this, combined with a coding error, resulted in the controller relying primarily on the PD terms of the control law for trajectory tracking. Therefore, the tests below are most representative of the performance of a PD controller. Simulation results from Chapter 4 indicate that the performance of the new controller will be much better, but hardware tests remain as future work.

Test 3, shown in Figure A.9 shows one of the many challenges of testing on an air-bearing floor as the satellite sticks in place until the final docking maneuver where a successful docking occurred. Figure A.10 shows a successful perpendicular docking for a demonstration of collocated docking. Figure A.11 shows an attempt at a complete assembly demonstration, resulting in a very near miss of the docking port. In this test the pin bounced off of the target docking port but failed to capture.

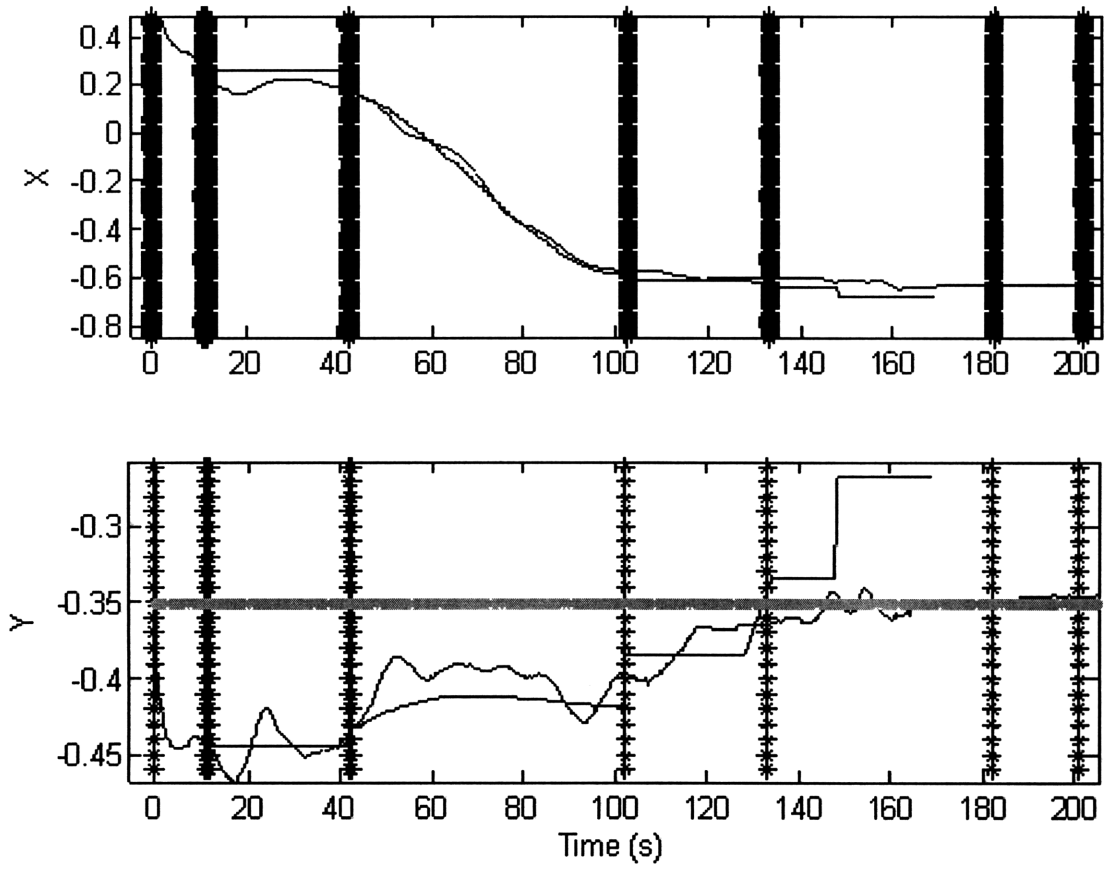


Figure A.8: Test 2: The satellite docks with the beam at approximately 160 seconds.

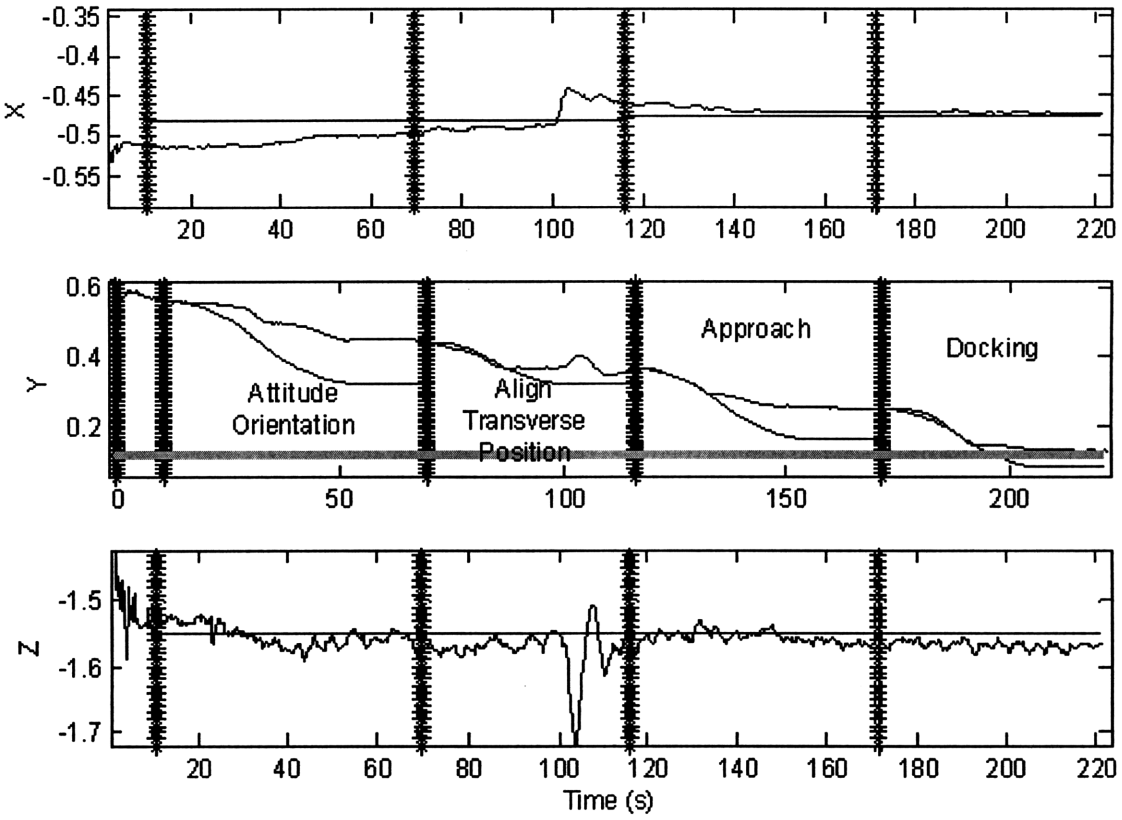


Figure A.9: Test 3: Satellite+Beam docking to fixed target occurs at approximately 200 seconds. The satellite clearly gets stuck to the floor at 130 seconds while following the Y trajectory.

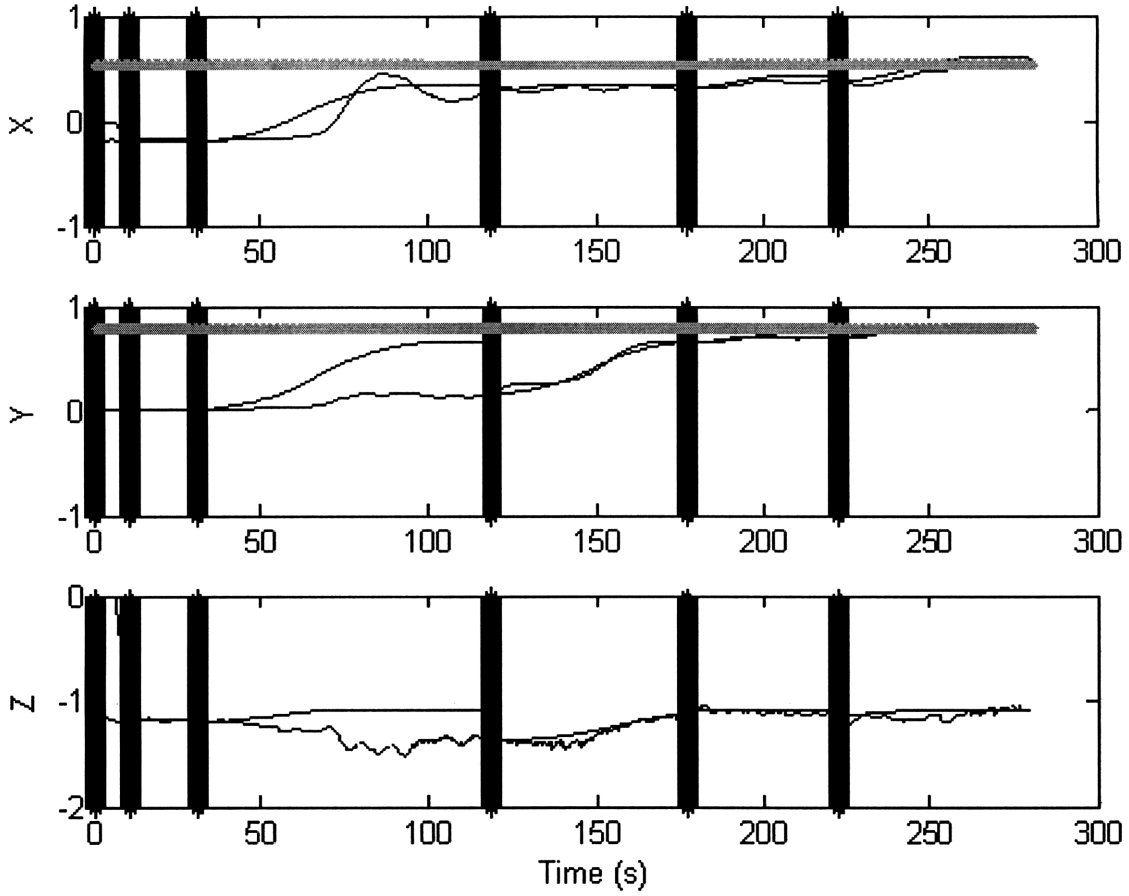


Figure A.10: Test 4: Satellite+Beam dock perpendicularly to a target at approximately 250 seconds.

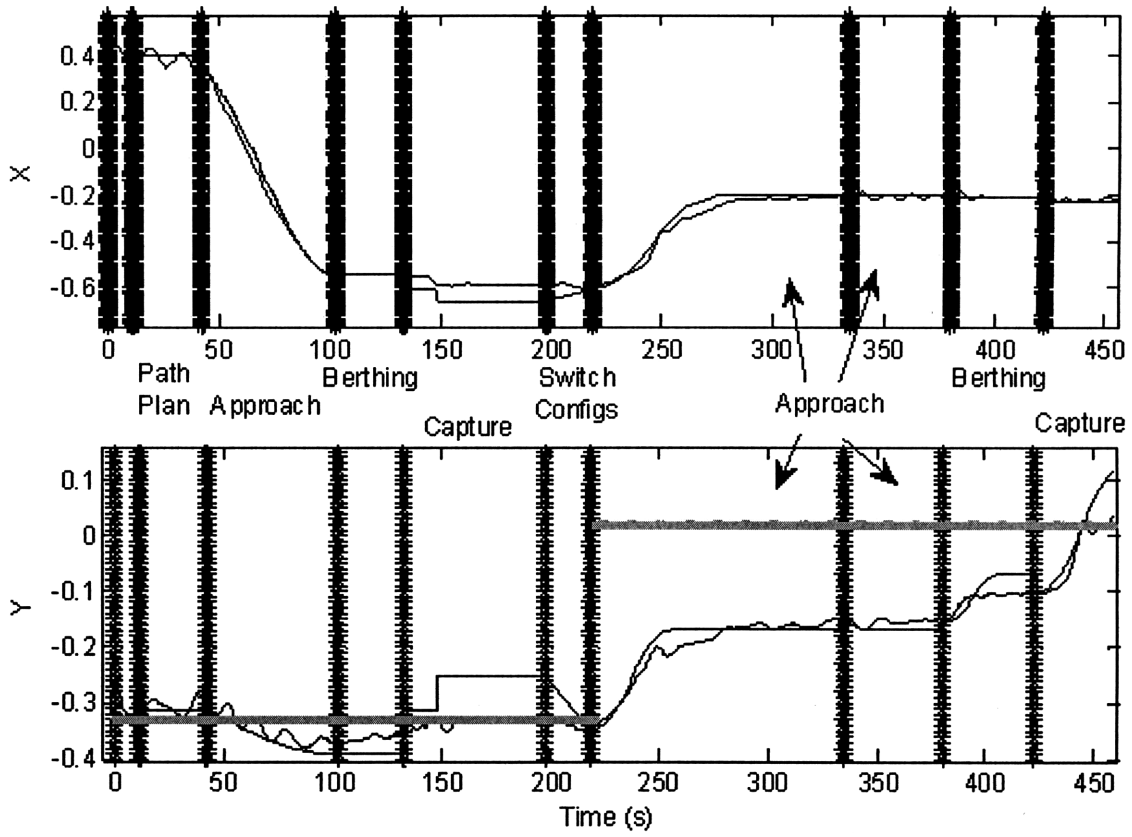


Figure A.11: The propulsion module docks with the beam at 150 seconds, then maneuvers the beam for a (missed) docking attempt at 425 seconds.

Appendix B

Scripts for Symbolic Robot Dynamics

B.1 EulerLagrange.m

```
function [H,C,G] = EulerLagrange(T,V,q)
% Creates the symbolic manipulator equations from the Lagrangian T-V
% Inputs:
%   T       Kinetic energy
%   V       Potential energy
%   q       vector of symbolic state variables (must be functions of time)
%
% Returns the coefficient matrices H, C, and G, for the left hand side of
% the standard nonlinear manipulator equations:
%   H(q)q_ddot + C(q,q_dot)q_dot + G(q) = Tau

%% Create additional symbolic variables
syms t;
Nstates = length(q);
for i = 1:Nstates
    qdum(i) = sym(['q' num2str(i) '_sym']);
    qddum(i) = sym(['q' num2str(i) 'd_sym']);
end
q_dot = diff(q,t);

%% Calculate H, C, and G

%Make a temporary copy of T and V
Ttemp = T;
Vtemp = V;
%Substitute functions of time for dummy variables DO VELOCITY FIRST!
for j = 1:length(q);
```

```
    Ttemp=subs(Ttemp,q_dot(j),qddum(j));
    Ttemp=subs(Ttemp,q(j),qdum(j));
    Vtemp=subs(Vtemp,q_dot(j),qddum(j));
    Vtemp=subs(Vtemp,q(j),qdum(j));
end

%Find H
%This is equivalent to partial^2/partial(qdot)^2 of T since the potential
%term of L is only dependent on q
H = jacobian(jacobian(Ttemp,qddum),qddum);
H=simple(H);
H=TrigSimple(H);

%Construct C explicitly from H
C = sym(zeros(length(q)));
for j = 1:length(q)
    for i = 1:length(q)
        for k = 1:length(q)
            C(i,j) = C(i,j)+0.5*diff(H(i,j),qdum(k))*qddum(k) + 0.5*(diff(H(i,k),qdum
                (j))-diff(H(j,k),qdum(i)))*qddum(k);
        end
    end
end
C=simple(C);

%Construct G from V
G = jacobian(Vtemp,qdum).';
G=simple(G);

%Replace functions of time
for j = 1:length(q);
    %Do positions first
    C = subs(C,qdum(j),q(j));
    H = subs(H,qdum(j),q(j));
    G = subs(G,qdum(j),q(j));
    C = subs(C,qddum(j),q_dot(j));
    H = subs(H,qddum(j),q_dot(j));
end
```

B.2 CreateSimFromDyn.m

```
function CreateSimFromDyn(filename,H,C,G,q,params,vals)
% Creates a file called 'filename'.m that simulates the dynamics specified
% in H, C, and G, for the generalized coordinates q.
%
```

```
% Inputs:
%   filename     desired name for the file
%   H            mass matrix (also known as M), H(q,qdot)
%   C            coriolis matrix, C(q,qdot)
%   G            potential terms, G(q)
%   params       symbolic parameters to be substituted
%   vals         values of symbolic parameters
%
% The generated file has the function signature:
%
%   function xdot = filename(x,u)
%
% where x is the current state, u is the input, and xdot is the state
% derivative.  The file is generated with a %#eml header so that it can be
% called from an EML block, or you can use ODE45 with a wrapper function.
%% Substitute for fcns of time
syms t;
qdot=diff(q,t);
Htemp = H;
Gtemp = G;
Ctemp = C;
%Iterate through state vector and replace (do vel. first!)
for i = 1:length(q)
    Htemp=subs(Htemp,qdot(i),['q' num2str(i) 'ds']);
    Htemp=subs(Htemp,q(i),['q' num2str(i) 's']);
    Ctemp=subs(Ctemp,qdot(i),['q' num2str(i) 'ds']);
    Ctemp=subs(Ctemp,q(i),['q' num2str(i) 's']);
    Gtemp=subs(Gtemp,qdot(i),['q' num2str(i) 'ds']);
    Gtemp=subs(Gtemp,q(i),['q' num2str(i) 's']);
end
syms t;

real_ary = SubstituteValues({Htemp,Ctemp,Gtemp},params,vals);

names = {'Hsim', 'Csim', 'Gsim'};
%% Write Code
disp(['Writing equations of motion to ' filename '.m'])
handle=fopen([filename '.m'],'w+');
header = ['%%%#eml\n' 'function xdot = ' filename '(x,u)\n'];

%Initialize matrices
qlen = num2str(length(q));
stlen = num2str(2*length(q));
header = [header 'Hsim = zeros(' qlen ',' qlen ');\n'];
header = [header 'Csim = zeros(' qlen ',' qlen ');\n'];
```

```
header = [header 'Gsim = zeros(' qlen ',1);\n'];
header = [header 'xdot = zeros(' stlen ',1);\n'];

%Write header
fprintf(handle,header);

%Print state variables
vars = [];
varcat = [];
vardcat = [];
for i = 1:length(q)
    idx = num2str(i);
    varstr = ['q' idx 's'];
    vardstr = ['q' idx 'ds'];
    d_idx = num2str(i+length(q));
    vars = [vars 'q' idx 's = x(' idx ');\n'];
    vars = [vars 'q' idx 'ds = x(' d_idx ');\n'];
    varcat = [varcat ' ' varstr];
    vardcat = [vardcat ' ' vardstr];
end
vars = [vars 'q=[' varcat ']'';\n'];
vars = [vars 'qd=[' vardcat ']'';'];

fprintf(handle,vars);

for i = 1:3
    %clean up weird substitution problem
    real_ary{i} = sym(real_ary{i});
    code = ccode(real_ary{i});
    code = strrep(code,'T',names{i});
    %change to matlab indices
    for k = 1:max(size(real_ary{i}))
        code=strrep(code,['[' num2str(k-1) ']'],['(' num2str(k) ')']);
        code=strrep(code,')(',',');
    end
    %add line breaks
    code=strrep(code,';',';\n');
    fprintf(handle,['\n' code]);
end

qdstart = num2str(length(q)+1);
qdran = [qdstart ':end'];
footer=['xdot = [x(' qdran '); Hsim\\(u - Gsim - Csim*qd)];'];

fprintf(handle,footer);
```

```
fclose(handle);
```

B.3 LinearizeBeam.m

```
function [Hlin,Clin,Glin] = LinearizeBeam(H,C,G,q,q_op,qd_op)
% Linearizes beam equations about q_op and qd_op so they are in the form
% H_lin*q_ddot + C_lin*qdot + G_lin*q

% Create q_dot vector
Nstates = length(q);
syms t;
q_dot = diff(q,t);

% Form G matrix (was a vector)
Glin = sym(zeros(Nstates));

% Substitute for states in C and H
Clin = C;
Hlin = H;
for i = 1:Nstates
    Clin=maple('eval',Clin,[char(q_dot(i)) '=' num2str(qd_op(i),100)]);
    Clin=maple('eval',Clin,[char(q(i)) '=' num2str(q_op(i),100)]);
    Hlin=maple('eval',Hlin,[char(q_dot(i)) '=' num2str(qd_op(i),100)]);
    Hlin=maple('eval',Hlin,[char(q(i)) '=' num2str(q_op(i),100)]);
    for j = 1:Nstates
        qcoeff = maple('coeff',G(j),q(i));
        Glin(i,j) = qcoeff;
    end
end
Hlin=simple(Hlin);
Clin=simple(sym(Clin));
```

B.4 SymbolicBeamDynamics.m

This file wraps the previous files into a single script to generate symbolic and linearized dynamics for the segmented beam model.

```
%% Symbolic Dynamics for the Segmented Beam
% This file generates the symbolic dynamics and linear dynamics for the
% flexible beam.  There are 5 major sections:
%
% 1. Builds symbolic variables and generates the Lagrangian
% 2. Calculates symbolic dynamics from the symbolic lagrangian
% 3. Generates a nonlinear beam simulation
```

```
% 4. Linearizes the beam about the zero state
% 5. Creates a linear state space system

%% 1. Define Symbolic Variables and Create Lagrangian
%State is defined:
% X = [x y q0 q1 q2 q3 xdot ydot q0dot q1dot q2dot q3dot]
%Input is defined:
% u = [Fx Fy Tau]
maple('restart');
syms l1s l2s l3s l4s t;
syms m_sphs m1s m2s m3s m4s m_dps;
syms r_swarms r_dps
syms I_sphs I1s I2s I3s I4s I_dps
syms k1s k2s k3s;

%Make angle velocities explicit functions of time
x = sym(maple('x(t)'));
y = sym(maple('y(t)'));
q0 = sym(maple('q0(t)'));
q1 = sym(maple('q1(t)'));
q2 = sym(maple('q2(t)'));
q3 = sym(maple('q3(t)'));
q = [x;y;q0;q1;q2;q3];
%Create simple expressions for sin and cos of common angles
s0 = sin(q0);
s01 = sin(q0+q1);
s012 = sin(q0+q1+q2);
s0123 = sin(q0+q1+q2+q3);

c0 = cos(q0);
c01 = cos(q0+q1);
c012 = cos(q0+q1+q2);
c0123 = cos(q0+q1+q2+q3);

%Create angular velocities of each link
w0 = diff(q0,t);
w1 = w0 + diff(q1,t);
w2 = w1 + diff(q2,t);
w3 = w2 + diff(q3,t);

%Create positions of centers of mass
x0 = [x;y];
x1 = x0 + (r_swarms+l1s/2)*[c0;s0];
x2 = x0 + (r_swarms+l1s)*[c0;s0] + l2s/2*[c01;s01];
x3 = x0 + (r_swarms+l1s)*[c0;s0] + l2s*[c01;s01] + l3s/2*[c012;s012];
```

```
x4 = x0 + (r_swarms+l1s)*[c0;s0] + l2s*[c01;s01] + l3s*[c012;s012]+l4s/2*[c0123;s0123
    ];
x5 = x0 + (r_swarms+l1s)*[c0;s0] + l2s*[c01;s01] + l3s*[c012;s012]+(l4s+r_dps/2)*[
    c0123;s0123]; %DP position

X = [x0 x1 x2 x3 x4 x5];
X_dot = diff(X,t);
M = [m_sphs; m1s; m2s; m3s; m4s; m_dps];
I = [I_sphs; I1s; I2s; I3s; I4s; I_dps];
omega = [w0;w0;w1;w2;w3;w3];

%Create Kinetic Energy
T=0;
for i=1:size(X,2);
    T = T+1/2*(M(i)*X_dot(:,i).'*X_dot(:,i) + omega(i)^2*I(i));
end

%Create potential energy function
V1 = 1/2*k1s*q1^2;
V2 = 1/2*k2s*q2^2;
V3 = 1/2*k3s*q3^2;
V = V1+V2+V3;

%Compute Lagrangian
Lagr = T-V;
%% 2. Create nonlinear symbolic state matrices
[Hnl,Cnl,Gnl]=EulerLagrange(T,V,q);
Hnl=TrigSimple(Hnl);
Cnl=TrigSimple(Cnl);
%% Param Vecs
sym_vec = [M.' I.' l1s l2s l3s l4s k1s k2s k3s r_swarms r_dps];
real_vec = [m_SWARM m0 m1 m2 m3 m_DP I_SWARM(3,3) I0(3,3) I1(3,3) I2(3,3) I3(3,3)
    I_DP(3,3) L(1) L(2) L(3) L(4) k1 k2 k3 r_SWARM r_DP];

%% 3. Generate a Beam Simulation (BeamEOM.m)
CreateSimFromDyn('BeamEOM',Hnl,Cnl,Gnl,q,sym_vec,real_vec);

%% 4. Linearize about 0
[Hlin,Clin,Glin]=LinearizeBeam(Hnl,Cnl,Gnl,q,zeros(6,1),zeros(6,1));

%% Substitute Actual Values

real_ary = SubstituteValues({Hlin,Clin,Glin},sym_vec,real_vec);
Hreal = real_ary{1};
Creal = real_ary{2};
```

```
Greal = real_ary{3};
clear real_ary;

%% 5. Create state space model with linearized dynamics
Ns = length(q); %number of position states
Ni = 3; %number of inputs
Alin = eval([zeros(Ns) eye(Ns); -Hreal\Greal -Hreal\Creal]);
Blin = eval([zeros(Ns,Ni); Hreal\[eye(Ni); zeros(Ns-Ni,Ni)]]);
plant = ss(Alin,Blin,eye(size(Alin)),0);
% create discrete model
plantd = c2d(plant,1/20);

%% Save dynamics
save beam_dynamics_short Alin Blin Hreal Creal Greal Hlin Clin Glin Hnl Cnl Gnl q
    plantd plant;
```

Bibliography

[1] A. Arisoy, M. Gokasan, and O. Bogosyan. Partial feedback linearization control of a single flexible link robot manipulator. *Recent Advances in Space Technologies, 2005. RAST 2005. Proceedings of 2nd International Conference on*, pages 282–287, 9-11 June 2005.

[2] C.C. Cheah, C. Liu, and J.J.E. Slotine. Adaptive jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models. *Automatic Control, IEEE Transactions on*, 51(6):1024–1029, June 2006.

[3] Allen Chen. Propulsion system characterization for the SPHERES formation flight and docking testbed. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2002.

[4] Soon-Jo Chung. *Nonlinear Control and Synchronization of Multiple Lagrangian Systems with Application to Tethered Formation Flight Spacecraft*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2007.

[5] A. de Luca and B. Siciliano. Regulation of flexible arms under gravity. *Robotics and Automation, IEEE Transactions on*, 9(4):463–467, Aug 1993.

[6] S. Dubowsky and P. Boning. The coordinated control of space robot teams for the on-orbit construction of large flexible space structures. In *Proceedings of the 2007 IEEE International Conference Robotics and Automation, Special Workshop on Space Robotics*, April 2007.

[7] S.S. Ge, T.H. Lee, and G. Zhu. Tip tracking control of a flexible manipulator using pd type controller. *Control Applications, 1996., Proceedings of the 1996 IEEE International Conference on*, pages 309–313, Sep 1996.

[8] F. Ghorbel, J.Y. Hung, and M.W. Spong. Adaptive control of flexible joint manipulators. pages 1188–1193 vol.2, May 1989.

[9] You-Liang Gu and Yangsheng Xu. A normal form augmentation approach to adaptive control of space robot systems. *Dynamics and Control*, 5(3):275–294, July 1995.

[10] An-Chyau Haung and Kuo-Kai Liao. Fat-based adaptive sliding control for flexible arms: Theory and experiments. *Journal of Sound and Vibration*, 298(1-2):194 – 205, 2006.

[11] Mark O. Hilstad. A multi-vehicle testbed and interface framework for the development and verification of separated spacecraft control algorithms. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2002.

[12] Nicholas Hoff. Design and implementation of a relative state estimator for docking and formation control of modular autonomous spacecraft. Master of science thesis, Massachusetts Institute of Technology, 2007.

[13] Berthold K. Horn. *Robot Vision*. McGraw-Hill Higher Education, 1986.

[14] Y. Ishijima, D. Tzeranis, and S. Dubowsky. On-Orbit Maneuvering of Large Space Flexible Structures by Free-Flying Robots. In *'i-SAIRAS 2005' - The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, volume 603 of *ESA Special Publication*, August 2005.

[15] V. Kapila, A. Tzes, and Qiguo Yan. Closed-loop input shaping for flexible structures using time-delay control. *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, 2:1561–1566 vol.2, 1999.

[16] Matthew D. Lichter, Hiroshi Ueno, and Steven Dubowsky. Vibration estimation of flexible space structures using range imaging sensors. *International Journal of Robotics Research*, 25, No. 10:1001–10012, 2006.

[17] Gunter Niemeyer and Jean-Jacques E. Slotine. Performance in adaptive manipulator control. *Int. J. Rob. Res.*, 10(2):149–161, 1991.

[18] Simon Nolet. *Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite*. Sc. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2007.

[19] Lennon Rodgers. Concepts and technology development for the autonomous assembly and reconfiguration of modular space systems. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, February 2006.

[20] Alvar Saenz-Otero. *Design Principles for the Development of Space Technology Maturation Laboratories Aboard the International Space Station*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2005.

[21] Robert M. Sanner and Jean-Jacques E. Slotine. Stable adaptive control of robot manipulators using "neural" networks. *Neural Comput.*, 7(4):753–790, 1995.

[22] M. Shahravi and M. Kabganian. Attitude tracking and vibration suppression of flexible spacecraft using implicit adaptive control law. *American Control Conference, 2005. Proceedings of the 2005*, pages 913–918 vol. 2, June 2005.

[23] Alexander Shkolnik and Russ Tedrake. High-dimensional underactuated motion planning via task space control. *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3762–3768, Sept. 2008.

[24] D. Simon and Tien Li Chia. Kalman filtering with state equality constraints. *Aerospace and Electronic Systems, IEEE Transactions on*, 38(1):128–136, 2002.

[25] A. Sinha and D. W Miller. Optimal sliding-mode control of a flexible spacecraft under stochastic disturbances. *AIAA Journal of Guidance, Control, and Dynamics*, 18(3):486–492, MAY-JUN 1995.

[26] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control.* Prentice Hall, Upper Saddle River, New Jersey, 1991.

[27] M.W. Spong. Partial feedback linearization of underactuated mechanical systems. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 314–321, Sep 1994.

[28] H. A. Talebi, R. V. Patel, and K. Khorasani. *Control of Flexible-Link Manipulators Using Neural Networks.* Springer-Verlag, London, UK, 2001.

[29] H. A. Talebi, R. V. Patel, and K. Khorasani. A neural network controller for a class of nonlinear non-minimum phase systems with application to a flexible-link manipulator. *Journal of Dynamic Systems, Measurement, and Control*, 127(2):289–294, 2005.

[30] G.D. Warshaw, H.M. Schwartz, and H. Asmer. Stability and performance of sampled-data robot adaptive controllers. *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, 1:95–104, Jul 1992.

[31] J.T. Wen and R. Buche. Modeling and control of a rotating flexible beam on a translated base. pages 1646–1651 vol.2, Dec 1991.

[32] Ahmet S. Yigit. On the stability of pd control for a two-link rigid-flexible manipulator. *Journal of Dynamic Systems, Measurement, and Control*, 116(2):208–215, 1994.