

3-2021

## Adaptive Augmentation of Non-Minimum Phase Flexible Aerospace Systems

Michael A. DuPuis  
michael.a.dupuis@nasa.gov

Follow this and additional works at: <https://commons.erau.edu/edt>



Part of the [Navigation, Guidance, Control and Dynamics Commons](#)

---

### Scholarly Commons Citation

DuPuis, Michael A., "Adaptive Augmentation of Non-Minimum Phase Flexible Aerospace Systems" (2021).  
*PhD Dissertations and Master's Theses*. 640.  
<https://commons.erau.edu/edt/640>

This Dissertation - Open Access is brought to you for free and open access by Scholarly Commons. It has been accepted for inclusion in PhD Dissertations and Master's Theses by an authorized administrator of Scholarly Commons. For more information, please contact [commons@erau.edu](mailto:commons@erau.edu).

Fall 2021

# ADAPTIVE AUGMENTATION OF NON-MINIMUM PHASE FLEXIBLE AEROSPACE SYSTEMS

Michael A. DuPuis

Follow this and additional works at: <https://commons.erau.edu/edt>



Part of the [Navigation, Guidance, Control and Dynamics Commons](#)

---

This Dissertation - Open Access is brought to you for free and open access by Scholarly Commons. It has been accepted for inclusion in PhD Dissertations and Master's Theses by an authorized administrator of Scholarly Commons. For more information, please contact [commons@erau.edu](mailto:commons@erau.edu).

ADAPTIVE AUGMENTATION OF NON-MINIMUM  
PHASE FLEXIBLE AEROSPACE SYSTEMS

By

Michael A. DuPuis

A Dissertation Submitted to the Faculty of Embry-Riddle Aeronautical University  
In Partial Fulfillment of the Requirements for the Degree of  
Doctor of Philosophy in Aerospace Engineering

March 2021

Embry-Riddle Aeronautical University

Daytona Beach, Florida

# ADAPTIVE AUGMENTATION OF NON-MINIMUM PHASE FLEXIBLE AEROSPACE SYSTEMS

By

Michael A. DuPuis

This Dissertation was prepared under the direction of the candidate's Dissertation Committee Chair, Dr. Mark Balas, Department of Aerospace Engineering, and has been approved by the members of the Dissertation Committee. It was submitted to the Office of the Senior Vice President for Academic Affairs and Provost, and was accepted in the partial fulfillment of the requirements for the Degree of Philosophy in Aerospace Engineering.

## DISSERTATION COMMITTEE

\_\_\_\_\_  
Chairperson, Dr. Mark Balas

\_\_\_\_\_  
Co-Chairperson, Dr. Richard Prazenica

\_\_\_\_\_  
Member, Dr. Hever Moncayo

\_\_\_\_\_  
Member, Dr. Sergey Drakunov

\_\_\_\_\_  
Member, Dr. Tannen VanZwieten

\_\_\_\_\_  
Graduate Program Coordinator,  
Dr. Sirish Namilae

\_\_\_\_\_  
Date

\_\_\_\_\_  
Dean of the College of Engineering,  
Dr. James Gregory

\_\_\_\_\_  
Date

\_\_\_\_\_  
Senior Vice President for Academic  
Affairs and Provost,  
Dr. Lon Moeller

\_\_\_\_\_  
Date

## ACKNOWLEDGEMENTS

*I would like to thank...*

### MY FAMILY

*Your love and support provide the foundation for every meaningful endeavor. I don't have words for what you mean to me.*

### DR. MARK BALAS

*For guiding not just this work, but indeed my academic and professional trajectory, with such extraordinary wisdom, patience, and grace.*

### DR. RICHARD PRAZENICA

*For your immense assistance as Co-Chair, and for all the teaching and guidance throughout this journey.*

### MY DISSERTATION COMMITTEE

*For all the encouragement, assistance, and wonderful advice along the way.*

### MY FRIENDS

*For not disowning me. We'll grab some wings and watch some ball, I promise.*

### MY INTERNS AND FELLOW STUDENTS

*It takes an army, and I've been blessed to have you as mine.*

### THE KENNEDY GRADUATE FELLOWSHIP PROGRAM

*For granting me the time, funding, and flexibility to realize this dream.*

*AD ASTRA PER ASPERA*

## ABSTRACT

This work demonstrates the efficacy of direct adaptive augmentation on a robotic flexible system as an analogue of a large flexible aerospace structure such as a launch vehicle or aircraft. To that end, a robot was constructed as a control system testbed. This robot, named “Penny,” contains the command and data acquisition capabilities necessary to influence and record system state data, including the flex states of its flexible structures. This robot was tested in two configurations, one with a vertically cantilevered flexible beam, and one with a flexible inverted pendulum (a flexible cart-pole system).

The physical system was then characterized so that linear analysis and control design could be performed. These characterizations resulted in linear and nonlinear models developed for each testing configuration. The linear models were used to design linear controllers to regulate the nominal plant’s dynamical states. These controllers were then augmented with direct adaptive output regulation and disturbance accommodation. To accomplish this, sensor blending was used to shape the output such that the nonminimum phase open loop plant appears to be minimum phase to the controller.

It was subsequently shown that augmenting linear controllers with direct adaptive output regulation and disturbance accommodation was effective in enhancing system performance and mitigating oscillation in the flexible structures through the system’s own actuation effort.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iii
ABSTRACT.....	iv
LIST OF FIGURES .....	vii
LIST OF TABLES.....	xii
SYMBOLS.....	xiii
ABBREVIATIONS .....	xiv
1. Introduction.....	1
1.1 Motivation.....	2
1.2 Introduction to Inverted Pendulum Systems.....	5
1.3 Direct Adaptive Control and Augmentation for Disturbances .....	7
2. Control and Augmentation.....	12
2.1. Full State Feedback and Separation Principle Control .....	12
2.2. Direct Model-Referencing Adaptive Control .....	13
2.3. Closed-loop Stability Analysis .....	18
2.4. Augmented Direct Adaptive Regulation and Disturbance Mitigation.....	20
2.5. Sensor Blending.....	21
3. Testbed Hardware .....	23
3.1 Cart System.....	23
3.2 Pendulum System.....	26
3.3 Configurations.....	27
3.4 Electrical Systems.....	29
3.4.1. Electrical Power Storage and Distribution .....	29
3.4.2. Command and Control.....	30
3.4.3. Instrumentation.....	30
4. Testbed Software .....	33
4.1. Software for Intra-robot Sensing and Communication.....	33
4.2. Penny Programming and Data Acquisition.....	33
5. Model Development.....	39
5.1. Oscillatory Dynamics Characterization .....	40
5.2. CFS Linear Model Development.....	42
5.3. FIP Linear Model Development .....	44
5.3.1. FIP Rigid Body Dynamics Modeling .....	45
5.3.2. Actuator Dynamics Model.....	46
5.3.3. Combined Rigid Body and Actuator Dynamics .....	49
5.3.4. FIP Flex Dynamics Modeling.....	52

5.3.5. FIP Disturbance State Modeling.....	56
5.4. Nonlinear Model Development.....	59
5.4.1. Simscape model development .....	59
5.5. FIP Linear Model Development from Non-Linear Simulation .....	63
6. Controller Development and Application on Hardware .....	67
6.1. CFS Rigid Body Control.....	67
6.2. Penny CFS Augmentation by Adaptive Regulation .....	73
6.3. Penny CFS Augmentation by Disturbance Accommodation.....	75
6.4. FIP Rigid Body Control.....	78
6.5. FIP Augmentation with Direct Adaptive Regulation.....	81
6.6. FIP Augmentation with Direct Adaptive Disturbance Accommodation .....	84
6.7. FIP Augmentation with Disturbance State Adaptive Accommodation .....	86
6.8. FIP Augmentation with Step Disturbance .....	88
7. Results and Future Work .....	91
7.1. Penny CFS Implementation .....	91
7.2. Penny FIP Implementation .....	92
7.3. Future Work.....	94
7.3.1. Sensor Blending.....	94
7.3.2. Output Disturbance Accommodation .....	95
7.3.3. Comparison with nonadaptive augmentation .....	95
7.3.4. Penny Hardware/Software.....	96
REFERENCES .....	98
LIST OF PUBLICATIONS .....	105
APPENDIX A: SUPPORTING MATHEMATICS.....	107
APPENDIX B: SOFTWARE .....	111

## LIST OF FIGURES

Figure	Page
1.1 The International Space Station is a collection of flexible structures such as trussing, modules, and solar arrays (NASA Image and Video Library, 2011)....	1
1.2 Ares I Modal Displacements (Jang, et al., August 2011). .....	3
1.3 Bode Plot example identifying showing gain and phase margins.....	3
1.4 Similarity between pole-cart and launch vehicle (Pei & Rothhaar, 2018). .....	5
1.5 Examples of (a) rigid (b) flexible inverted pendula with lateral cart actuation. ..	6
2.1 Direct adaptive state regulation and disturbance accommodation.....	21
2.2 Direct adaptive state regulation and disturbance accommodation.....	21
3.1 Penny cart CAD model: isometric view and maximum outer dimensions. ....	23
3.2 Upper Penny CAD model view with significant components identified.....	24
3.3 Lower Penny CAD model view with significant components identified. ....	24
3.4 Penny pendulum system hardware on Cart.....	27
3.5 Penny in (a) CFS and (b) FIP configurations.....	28
3.6 Penny Cart power distribution diagram. ....	29
3.7 Cart instrumentation schematic.....	31
3.8 Tip instrumentation mounted on FIP and CFS (inset) configurations.....	31
3.9 Tip instrumentation wiring schematic. ....	32
4.1 Simulink command and control programming interface for Penny FIP.....	34
4.2 Inside the Penny Plant subsystem block. ....	35
4.3 Motor commanding and wheel encoder reading subsystem. ....	35
4.4 Acquisition of $x$ and $\dot{x}$ states. ....	36
4.5 Acquisition of $\theta_b$ and $\dot{\theta}_b$ states from tip IMU. ....	36

Figure	Page
4.6	Decoding of the tip angle IMU's 12-bit signal. .... 37
4.7	Command signal builder for pendulum reset servos..... 37
4.8	Acquisition of $\theta_t$ and $\dot{\theta}_t$ states. .... 37
4.9	State measuring and recording..... 38
5.1	Recorded CFS (left) and FIP (right) angle data. .... 40
5.2	Simulink model for parameter ID data collection..... 42
5.3	Simulink model to create linearized state data from transfer functions..... 43
5.4	CFS Linear Model vs. Hardware for same input. .... 44
5.5	Block diagram of interaction between actuator and plant dynamics. .... 49
5.6	FIP Actuator and Rigid Body combined dynamics ..... 50
5.7	Linear Model vs. Hardware Data for Same Input Command, $u = 20$ ..... 51
5.8	Linear Model vs. Hardware Data for Same Input Command, $u = 10$ ..... 51
5.9	Schematic for flexible inverted pendulum model. .... 52
5.10	Rigid body and flexible dynamics model ..... 53
5.11	Simulink model for linearized system. .... 54
5.12	Model for combined system dynamics ..... 55
5.13	FIP combined linear model response to doublet disturbance. .... 56
5.14	Penny FIP “pluck” test showing angular states for pendulum base and tip..... 56
5.15	Penny FIP “pluck” test showing disturbance state. .... 57
5.16	State-space model of FIP with disturbance state, compared to FIP..... 58
5.17	Nonlinear Simulink cart-pole model under linear control. .... 59
5.18	Simscape actuator dynamics model..... 60

Figure	Page
5.19 Simulink Motor Signal Builder mapping controller output to motor voltage. ...	60
5.20 Simscape nonlinear dynamics model of CFS .....	61
5.21 Simscape nonlinear dynamics model of FIP.....	61
5.22 Resulting linear model vs. Simscape model for CFS. ....	62
5.23 Comparison of $\theta b$ state, simulation vs. FIP.....	63
5.24 Resulting Linear Rigid Body Model vs. Simscape Model for FIP .....	65
5.25 x-states comparison between Linear Rigid Body Model and Penny FIP. ....	65
5.26 Theta-states comparison between Linear Rigid Body Model and Penny FIP. ....	66
6.1 Penny CFS linear control with separation principle observer-controller.....	69
6.2 Simulink CFS model with separation principle observer-controller. ....	70
6.3 Controller/observer state responses to step input.....	71
6.4 Beam flex state responses to step input. ....	71
6.5 CFS control model with “unseen” disturbance.....	72
6.6 Cart state response to “unseen” disturbance. ....	72
6.7 Schematic of augmentation for adaptive regulation. ....	73
6.8 Simulink CFS control model augmented with adaptive regulation. ....	73
6.9 Penny CFS under unaugmented (red) and augmented (blue) control.....	74
6.10 Cumulative effort, unaugmented and augmented control (equation inset).....	75
6.11 Augmentation for adaptive regulation and disturbance accommodation. ....	76
6.12 Penny CFS with adaptive regulation and disturbance accommodation.....	76
6.13 Penny CFS measured states under unaugmented (red) and augmented (blue) adaptive regulation and sinusoidal disturbance accommodation. ....	77

Figure	Page
6.14 Cumulative effort, unaugmented and augmented control (equation inset).....	77
6.15 Diagram of Penny FIP system under linear control.....	79
6.16 Penny FIP control model with linear controller and excitation. ....	79
6.17 Penny FIP rigid body measured states under linear control (10 sec).....	80
6.18 Penny FIP rigid body measured states under linear control (3 sec).....	80
6.19 Penny FIP linear control model with flex present. ....	81
6.20 Penny FIP with rigid plant augmented with adaptive regulation.....	81
6.21 Penny FIP control model augmented with adaptive regulation. ....	82
6.22 Penny FIP angular states for rigid plant augmented with adaptive regulation ....	82
6.23 Penny FIP states under linear control with flex present. ....	83
6.24 Penny FIP augmented with adaptive regulation. ....	84
6.25 Penny FIP augmented with adaptive regulation. ....	84
6.26 Penny FIP with adaptive regulation and disturbance accommodation. ....	85
6.27 Penny FIP angular state comparison with and without augmentation.....	85
6.28 Penny FIP with adaptive augmentation. ....	86
6.29 Penny FIP augmented with adaptive regulation including disturbance state. ....	87
6.30 Penny FIP augmented vs. unaugmented states, $\sigma = 30$ . ....	87
6.31 Penny FIP augmented vs. unaugmented disturbance states, $\sigma = 30$ .....	88
6.32 Penny FIP with step disturbance, unaugmented. ....	89
6.33 FIP with step disturbance augmented with adaptive state regulation. ....	90
B.4.1 Motor Signal Builder.....	124
B.4.2 Piecewise function showing mapping $f: cmd \rightarrow ctrl$ .....	124

Figure	Page
B.4.3 Tuned motor signal builder. Yellow: cmd, purple & green: ctrl, red & blue: motor encoder readings.....	126

## LIST OF TABLES

Table	Page
3.1 Drive Train Hardware .....	25
3.2 Power Storage and Distribution Main Components .....	30
3.3 Instrumentation Components .....	32
5.1 Flexible structure oscillator parameters .....	42
5.2 Summary of State Variables .....	45
5.3 Rigid body parameters of Penny FIP .....	46
5.4 Summary of Actuator State Variables .....	46
5.5 Actuator Parameters.....	47

## SYMBOLS

$\forall$	for all
$\exists$	there exists
$\ni$	such that
$a$	math font; indicates a scalar quantity
$\dot{z}$	first time derivative of argument $z$
$\ddot{z}$	second time derivative of argument $z$
$x$	cart position
$\dot{x}$	cart velocity
$\theta$	rigid pendulum angle with respect to (w.r.t.) vertical
$\dot{\theta}$	rigid pendulum angular velocity w.r.t. vertical
$\theta_b$	flexible pendulum angle w.r.t. vertical at base of pendulum
$\dot{\theta}_b$	flexible pendulum angular velocity w.r.t. vertical at base of pendulum
$\theta_t$	flexible pendulum angle w.r.t. vertical at tip of structure
$\dot{\theta}_t$	flexible pendulum angular velocity w.r.t. vertical at tip of structure
$I_n$	the Identity Matrix of Dimension $n$
$a e^X$	$a 10^X$ where $a$ is a scalar and $X$ is an integer
$A$	state evolution matrix of a state-space system
$B$	input matrix of a state-space system
$C$	output matrix of a state-space system
$D$	feedthrough matrix of a state-space system

## ABBREVIATIONS

AAC	Adaptive Augmenting Control
AHRS	Attitude Heading Reference System
CFS	Cantilevered Flexible Structure configuration
DOAC	Direct Output Adaptive Control
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
ERAU	Embry-Riddle Aeronautical University
FIP	Flexible Inverted Pendulum configuration
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
IP	Inverted Pendulum
LQR	Linear Quadratic Regulator
MEMS	Microelectromechanical Systems
MRDAC	Model Reference Direct Adaptive Control
NASA	National Aeronautics and Space Administration
ODE	Ordinary Differential Equation
PID	Proportional, Integral, Derivative
PWM	Pulse Width Modulated
SLS	Space Launch System
UI	User Interface
USAF	United States Air Force

## 1. Introduction

Aerospace structures such as aircraft wings and fuselage components, space launch vehicles, and space station photovoltaic array trussing, tend to be flexible since mass is minimized to optimize flight system performance. But flexible aerospace structures are also susceptible to oscillation dynamics in the presence of disturbances. These disturbances may be the result of the system's internal dynamics through the excitation of oscillatory modes (flexible modes) by the system's own actuators. They may also be the result of external forces such as aerodynamic loading, solar radiation impingement, astronaut movement, and a host of other momentum-transferring processes.

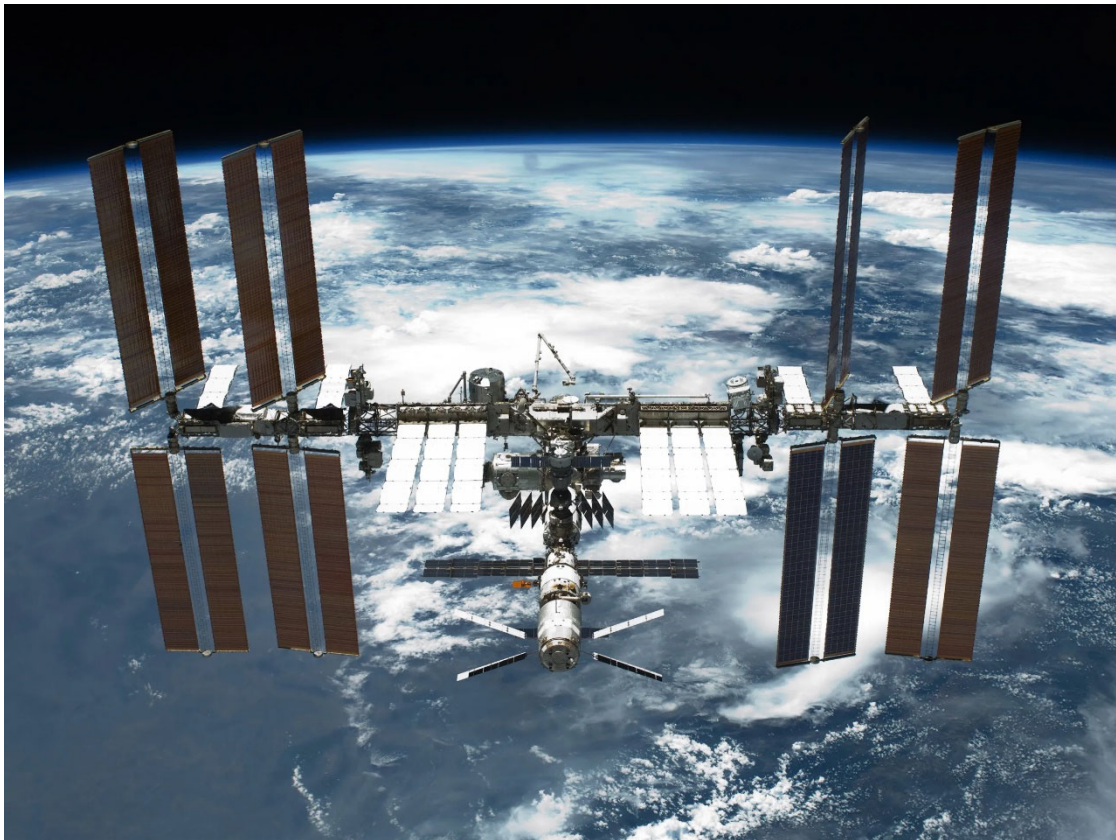


Figure 1.1 The International Space Station is a collection of flexible structures such as trussing, modules, and solar arrays (NASA Image and Video Library, 2011).

## 1.1 Motivation

Oscillatory behavior in a flexible aerospace system is usually deleterious. At low amplitude, oscillation may simply result in poor system performance, e.g., vibrating a system which relies on high precision pointing such as deep space imaging or long-range point-to-point communication. At high amplitude, a host of critical problems may arise such as reduced flight system stability margins or even structural failure resulting in total system loss (NESC, 2016). Adequate system performance may rely, therefore, on mitigating the system's oscillatory response to excitation. This is typically accomplished by implementing one or more mitigation techniques, all with their own benefits, drawbacks, and relative applicability based on system requirements. Common mitigation techniques in the aerospace industry include:

1. Aerospace structures may be designed such that their natural frequencies (e.g., Figure 1.2) reside far from known excitation frequencies.
2. Vibration attenuation elements may be built into the system to remove energy from excited flexible modes, much like an automobile's shock absorbers removing energy from the suspension system's spring-mass oscillator system.
3. Filters may be applied to state measurements based on known resonant frequencies to prevent oscillatory signals from feeding back into the control system.
4. Filters may be applied to the actuation signal to ensure that flex modes are not excited by the actuators themselves.
5. State oscillation may be measured (or estimated) and subsequently cancelled by actuation effort.

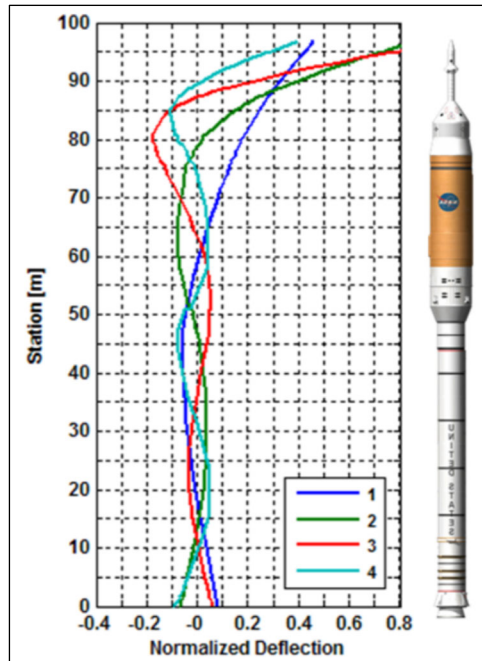


Figure 1.2 Ares I Modal Displacements (*Jang, et al., August 2011*).

Even with advances in non-linear control theory in recent decades, aerospace control system architectures remain primarily based on linear control theory. Some reasons for this continued prevalence are their ease of implementation, intuitive behavior, and strong theoretical underpinnings which result in objective performance metrics (e.g., Figure 1.3) such as gain and phase margins (Orr & VanZwieten, 2012).

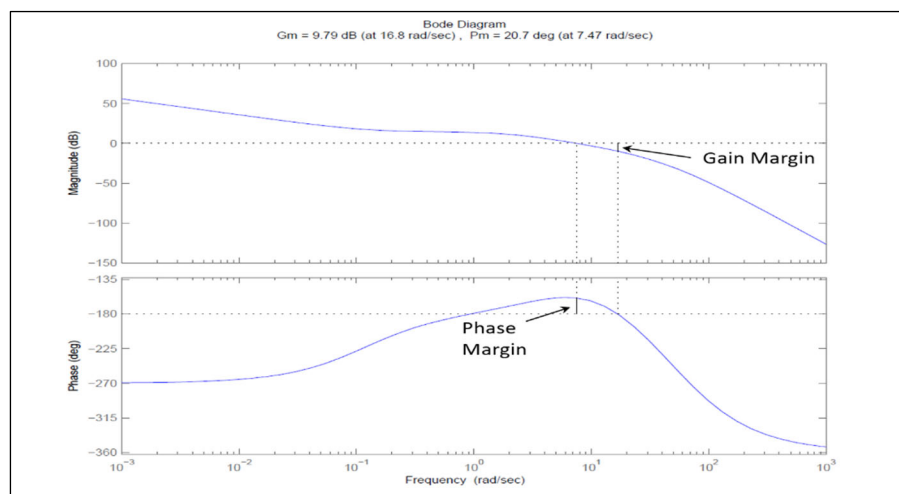


Figure 1.3 Bode Plot example identifying showing gain and phase margins.

Another reason linear controllers, such as Proportional-Integral-Derivative (PID), remain the gold standard is that, often, years or decades of development have gone into existing control system development, which has resulted in satisfactory performance. In these cases, there is often considerable institutional knowledge and intuition supporting their design and implementation, and an understandable reluctance to stray from tried-and-true methods.

Linear controllers do have weaknesses, especially when applied to flexible aerospace structures which are intrinsically nonlinear dynamical systems. There are many approaches to linear analysis, based on a real nonlinear flexible system. The most common approach to modeling elasticity, which offers many analytical conveniences (Greensite, 1967) is the superposition of elastic modes with the nonlinear rigid body dynamics (Orr J. , 2011). To be subsequently modeled as a linear time-invariant (LTI) system for controller design and application still requires the linearization of dynamical processes around points of operation (Ogata, 2002). Therefore, the ability of linear analysis and control methods to produce accurate performance metrics and achieve satisfactory state regulation depends on how accurately the linearized system model represents the real nonlinear system in the vicinity of each linearization point (Tobbe, Matras, & Wilson, 2009).

To retain the original linear controller yet improve control system performance in the presence of disturbances and unmodeled dynamics, the approach taken here is to augment the existing control system architecture in such a way as to retain linear controller performance near equilibrium, yet nonlinearly increase actuation effort as state divergence increases. This study investigates the application of direct adaptive control

for this purpose and does so by implementation on a robotic inverted pendulum cart-pole system.

## 1.2 Introduction to Inverted Pendulum Systems

Inverted pendula are unstable mechanical systems which have been utilized for the study of control system implementation since the 1950s (Åström & Furuta, 2000). They have also been used to model flight control of space launch vehicles in the initial and latter stages of flight, when aerodynamic forces are too small for aerodynamic stability (Lundberg & Barton, 2010).

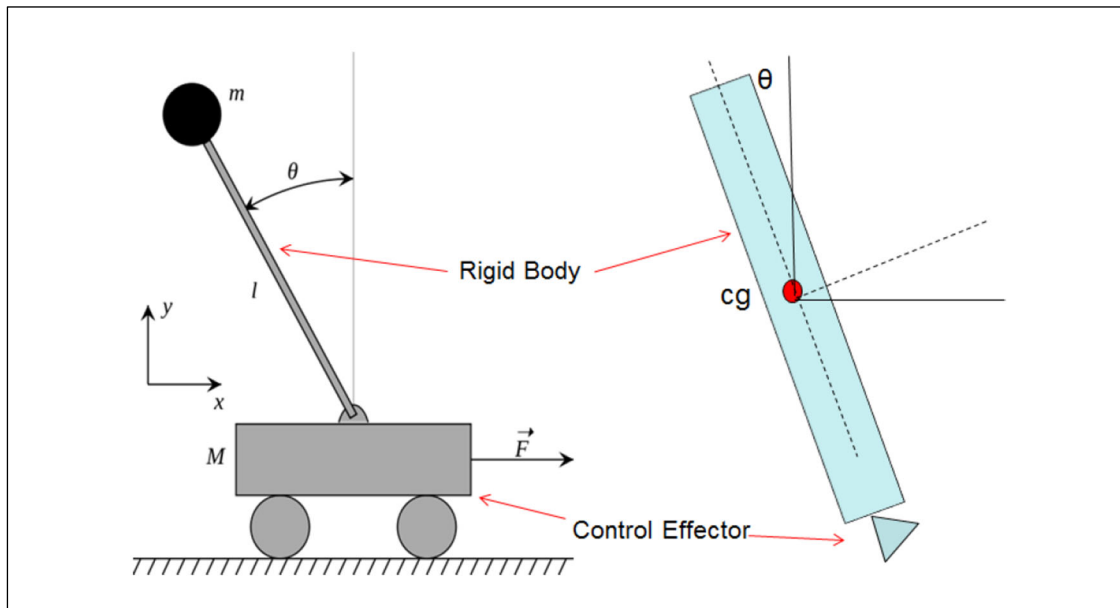


Figure 1.4 Similarity between pole-cart and launch vehicle (Pei & Rothhaar, 2018).

Many control architectures have been applied to the inverted pendulum problem throughout the years with examples far too numerous to exhaustively cover here. However, some notable examples organized by control system architecture are: model predictive-based (Magni, Scattolini, & Åström, 2002; Magni & Scattolini, 2004), H-infinity-based (Katayama, Yubai, & Hirai, 2009; Rigatos, Siano, Abbaszadeh, Ademi, &

Melkikh, 2017), neural network-based (Anderson, 1989), fuzzy logic-based (Wang, Tanaka, & Griffin, 1996), sliding mode-based (Park & Chwa, 2009), time optimal (Y. Xu & Furuta, 2001; Holzhüter, 2004), feed forward-based Mazenc & Praly, 2000; Mazenc & Bowong, 2003), energy-based (Åström & Furuta, 2000; Siuka & Schoberl, 2009), and adaptive-based (Åström & Wittenmark, 1995; Pei & Rothhaar, 2018), where a rigid inverted pendulum under LQR control is adaptively augmented for improved stability and reference signal tracking. For a more comprehensive treatment of inverted pendulum applications for control system research, the author commends to the interested reader's attention (Lundberg & Barton, 2010; Boubaker & Iriarte, 2017).

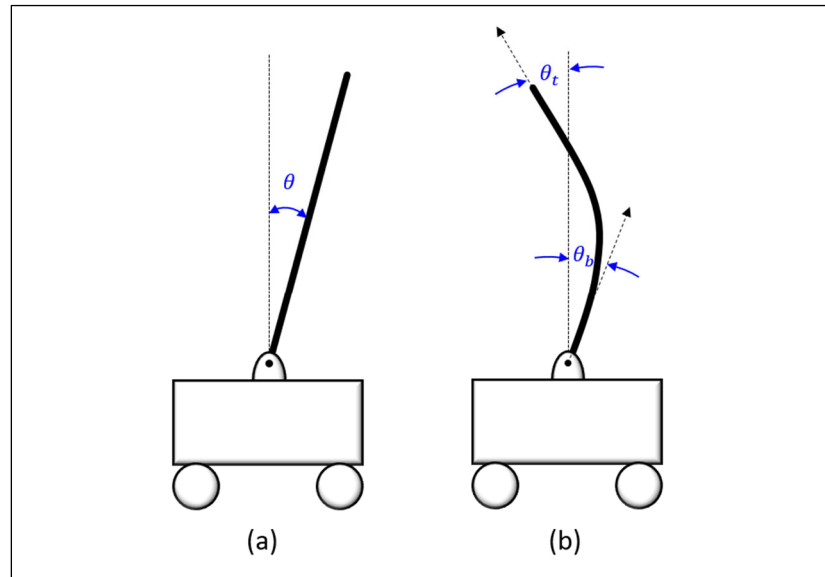


Figure 1.5 Examples of (a) rigid (b) flexible inverted pendula with lateral cart actuation.

Flexible structures have recently gained considerable attention owing to their high strength to weight ratios, especially in the aerospace and micromechanical systems' industries (Rahman & Isa, 2010). Rigid and flexible structures not only exhibit different dynamics, but they require different state descriptions and means of measurement. In the

case of rigid structures, the state variables are often readily obtained using rotary encoders, potentiometers, etc. However, for the flexible structures, the state variables also include elastic deformations and their velocities (Yoshikawa, Ohta, & Kanaoka, 2001), which are certainly more challenging to measure and model. This issue will come up later as flexible structures are modeled, and their states estimated and regulated.

To implement the theoretical work described herein, a wheeled mobile robot (“Penny”) is developed to accommodate control system testing with flexible structures including a fixed vertically cantilevered beam and a swiveling inverted pendulum. Sufficient instrumentation is included to obtain the requisite state information. MATLAB-based Simscape models are developed in addition to conventional ODE-based math models with state-space/transfer function representations. These models are essential for system analysis and controller/estimator design and testing before implementation on Penny.

### **1.3 Direct Adaptive Control and Augmentation for Disturbances**

Interest in controlling unmodeled plant dynamics began in earnest in the 1950’s as the USAF began testing high-performance aircraft. The dynamics of these vehicles changed so dramatically throughout their flight envelopes that active automated stabilization by the adaptation of flight controller gains was the subject of intense study at the time (Ioannou & Sun, 1996).

Model reference adaptive control (RMAC) was first proposed in (Whitaker, 1959; Osborn, Whitaker, & Kezer, 1961), yet lacked rigorous nonlinear stability analysis since (Lyapunov, 1892) was largely unknown in the West at the time. The ensuing three decades saw many seminal advancements in MRAC theory by such luminaries as

Monopoli, Narendra, Landau, Annaswami, Goodwin, Åström, Ioannou, Ortega, Sobel, Balas, Barkana, and many others, which included the formalization of stability criteria, largely based on the theories of Lyapunov.

At its essence, the goal of the MRAC approach seeks to make an ill-behaved dynamical system (the “plant”), in this case modeled as the linear time-invariant system

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad (1.1)$$

act like a well-behaved reference model:

$$\begin{cases} \dot{x}_m = A_m x_m + B_m u_m \\ y_m = C_m x_m; \quad x_m(0) = x_0 \end{cases} \quad (1.2)$$

If the original plant parameters are known, the gains that comprise the control signal  $u(t)$  can be designed to force the plant state vector  $x(t)$  to track  $x_m(t)$  and consequently the plant output vector  $y(t)$  tracks the reference model output vector  $y_m(t)$ . In the case when some plant parameters are unknown (which is nearly always the case for real systems), a tracking error state is defined as the difference between the plant and model outputs

$$e_y(t) \equiv y_p(t) - y_m(t) \quad (1.3)$$

The idea is that there is an adaptive gain for the  $i$ th parameter defined as

$$\dot{g}_{xi}(t) \equiv \sigma_i e_y(t) x_{mi}(t) \quad (1.4)$$

where the parameter  $\sigma_i$  drives the tracking error to zero and the adaptive gain then remains constant. The control law now takes the general form

$$\begin{cases} u(t) = \sum g_{xi}(t) x_{mi}(t) = G_x(t) x_m(t), \\ \dot{G} = \sigma_i e_y(t) x_{mi}(t) \end{cases} \quad (1.5)$$

Rigorous proofs determined that such a control architecture drives both tracking error and state error to zero. However, the stability of the adaptive control could only be proven if the initial LTI plant's transfer function was Strictly Positive Real (SPR).

The idea of positive realness or “passivity” was first introduced in (Popov V. M., 1962) for dynamical systems and in (Kalman, 1964) for system optimality. Given the square LTI system  $(A, B, C)$ , its transfer function  $T(s) = C(sI - A)^{-1}B$  is called strictly positive real if there exists two positive definite symmetric matrices,  $P$  and  $Q$ , such that the following are satisfied:

$$\begin{cases} PA + A^T P = -Q \\ PB = C^T \end{cases} \quad (1.6)$$

It was soon shown that (1.6) implies that the system is minimum phase, meaning all roots of the transfer function numerator reside to the left of the  $j\omega$ -axis, and that the system's high-frequency gain,  $CB$ , is symmetric positive definite (Ioannou & Tao, 1987).

The relatively restrictive SPR condition was subsequently loosened in (Barkana & Kaufman, 1985), which defined Almost Strictly Positive Real (ASPR) systems as those for whom there exist a constant output feedback gain,  $\bar{G}_e$  such that the closed-loop system in (1.7) is SPR:

$$A_c = A - B\bar{G}_e C \quad (1.7)$$

Therefore, although the original system is not SPR, if there exists two positive definite symmetric matrices,  $P$  and  $Q$  such that the following are satisfied:

$$\begin{cases} P(A - B\bar{G}_e C) + (A - B\bar{G}_e C)^T P = -Q \\ PB = C^T \end{cases} \quad (1.8)$$

then the system is said to be ASPR and the system is stabilizable in feedback via the positive definite gain  $\bar{G}_e$  (Fradkov, 1976). Indeed, it was Fradkov who first published the

use of output feedback in the MRAC control law, later giving rise to the “Direct-MRAC” or DMRAC moniker.

A particular deficiency of MRAC up to this point was the fact that the reference model needed to be of the order of the plant. In 1978, the Command Generator Tracker (CGT), was formulated to address this issue. The CGT model does not reproduce the plant, only its desired input-output behavior with sufficient detail to generate the desired trajectory (Broussard & Berry, 1978).

Researchers soon began expanding DMRAC theory to accommodate large, flexible structural systems (Kaufman, Balas, Bar-Kana, & Rao, 1981) and (Barkana, Kaufman, & Balas, 1983), leading to its theoretical extension to infinite-dimensional systems (Wen & Balas, 1989), including discrete-time formulations (Balas M. , 1995).

Advances were also being made in the area of disturbance mitigation. Model tracking and disturbance rejection were incorporated into a unified architecture under the Internal Model Principle (Francis & Wonhan, 1975), where a differential equation describing the disturbance was included. At the same time, C.D. Johnson developed a method of disturbance accommodation, also using an ODE description of the disturbance, in terms of full-state feedback and estimation where the stability analysis is a consequence of the separation principle (Johnson, 1976). Accommodation of persistent disturbances was extended to adaptive control techniques in (Balas, Erwin, & Fuentes, 2000), and to robust DMRAC (Fuentes & Balas, 2002). Disturbance accommodation would also be addressed for infinite-dimensional (distributed) systems in (Balas & Frost, 2016). Recent advances have been made in adaptive augmentation for stabilization of infinite-dimensional

systems under linear fixed-gain control (Balas & Frost, 2018), which was also extended to nonminimum phase distributed systems (Balas & Frost, 2019).

## 2. Control and Augmentation

Here we introduce the vast topic of direct adaptive control. We will then narrow our focus to those elements of direct adaptive control most applicable to the stabilization of flexible aerospace structures. Finally, we will lay the groundwork for how these architectures might be applied to real physical systems already under linear control, such as a launch vehicle or, in our case, the Inverted Pendulum Robot – Penny.

### 2.1. Full State Feedback and Separation Principle Control

Where linear control is implemented and when it is reasonable to assume all dynamical states are available to the controller, full state feedback may be employed through direct measurement, or some states may be provided by an observer. If all states are known and available at any time, given the linear plant model

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \quad (2.1)$$

if  $(A, B)$  *Controllable* then  $\exists G \ni u = Gx$  results in  $(A + BG)$  with arbitrary poles, allowing the control designer to place the system's closed loop poles wherever desired (Kimura, 1975).

When all states ( $x$ ) are not known, a state estimator may be designed and implemented along with the controller  $G$ . The state estimator is designed such that:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y}) \\ \hat{y} = C\hat{x} + Du \end{cases} \quad (2.2)$$

with the goal of driving the error state to zero:

$$e \equiv (\hat{x} - x) \xrightarrow[t \rightarrow \infty]{} 0 \quad (2.3)$$

The systems error dynamics are now written as:

$$\dot{e} = (A - KC)e, \quad (2.4)$$

where the eigenvalues of  $(A - KC)$  must have negative real parts to guarantee convergence of the state estimator. If  $(A, C)$  *Observable* then  $\exists K \ni (A - KC)$  has arbitrary poles, allowing the designer to place the system's closed loop observer poles wherever desired (Kimura, 1975). These two theorems are combined in the separation principle with feedback control law  $u = G\hat{x}$  operating on the plant modeled by (2.1) with state estimates from (2.2). This combination of systems allows arbitrary controller/observer pole placement  $\Leftrightarrow (A, B)$  controllable and  $(A, C)$  observable.

## 2.2. Direct Model-Referencing Adaptive Control

This section presents the framework for the control approach under study, Direct Model-Reference Adaptive Control, with further applications for disturbance mitigation. The foundational approach presented here will be later tailored for various applications in simulation and on the robot. The following derivations can be found in various forms in the literature, e.g., (NESC, 2016) and (Frost, Balas, & Wright, 2009)

Consider the linear, time-invariant, finite-dimensional system:

$$\begin{cases} \dot{x}_p = Ax_p + Bu_p + \Gamma u_D \\ y_p = Cx_p; \quad x_p(0) = x_0 \end{cases} \quad (2.5)$$

where the plant state,  $x_p(t)$ , is an  $N$ -dimensional vector with  $M$ -dimensional control input vector,  $u_p(t)$ , and  $M$ -dimensional sensor output vector,  $y_p(t)$ ; therefore, the plant is *square*. The disturbance input vector,  $u_D(t)$ , is  $MD$ -dimensional and is modeled by Disturbance Generator:

$$\begin{cases} u_D = \Theta z_D \\ \dot{z}_D = Fz_D; \quad z_D(0) = z_0 \end{cases} \quad (2.6)$$

where the disturbance state,  $z_D(t)$ , is ND-dimensional. Such descriptions of persistent disturbances were first used in (Johnson, 1976) to describe signals of known form but unknown amplitude.

Equation (2.6) can be rewritten in the following equivalent form:

$$\begin{cases} u_D = \theta z_D \\ z_D = L\varphi_D \end{cases} \quad (2.7)$$

where  $\varphi_D$  is a vector composed of the known basis functions which make up the disturbance, and L is a matrix of dimension  $N_D \times \dim(\varphi_D)$ . Therefore, all elements of  $u_D$  exist in the disturbance space  $\text{span}\{\varphi_1, \varphi_2, \dots, \varphi_{nD}\}$  and  $\varphi_D \equiv [\varphi_1, \varphi_2, \dots, \varphi_{nD}]^T$ . The direct adaptive controller is designed to mitigate disturbances of this form. For example, Equation (2.8) describes a sinusoidal disturbance of known frequency,  $\omega_D$ , but unknown amplitude and phase.

$$\begin{cases} \theta = [1 \quad 0] \\ F = \begin{bmatrix} 0 & 1 \\ -\omega_D^2 & 0 \end{bmatrix} \end{cases} \quad (2.8)$$

Of course, if the plant and disturbance generator parameter matrices are known, a Separation Principle-based estimator/controller can be developed to manage plant states and suppress persistent disturbances via feedback. In this formulation it is not assumed that the plant and disturbance generator parameter matrices are known. It is, however, assumed that the disturbance basis vector  $\varphi_D$  is known, which is reasonable for our purposes, since the frequency of Penny's oscillatory modes are readily obtained through direct measurement and analysis (reference Section 5.1).

The control objective will now be to cause the system output,  $y(t)$ , to asymptotically track the output of a known reference model:

$$\begin{cases} \dot{x}_m = A_m x_m + B_m u_m \\ y_m = C_m x_m; x_m(0) = x_0 \end{cases} \quad (2.9)$$

where the reference model state,  $x_m(t)$ , is an  $N_m$ -dimensional vector. The reference model output,  $y_m(t)$ , has the same dimension as the plant output,  $y(t)$ . Actuation of the reference model is accomplished by  $u_m(t)$ , which is generated by

$$\dot{u}_m = F_m u_m; u_m(0) = u_0^m \quad (2.10)$$

The reference model parameters ( $A_m, B_m, C_m, F_m$ ) are assumed known.

Using the process described in (Wen & Balas, 1989), “Ideal Trajectories” are defined for the plant given by (2.9) as linear combinations of the reference model states, the control inputs, and the disturbance inputs:

$$\begin{cases} x_* = S_{11}^* x_m + S_{12}^* u_m + S_{13}^* z_D \\ u_* = S_{21}^* x_m + S_{22}^* u_m + S_{23}^* z_D \end{cases} \quad (2.11)$$

where the *Ideal Trajectory*  $x_*(t)$  is produced by the *Ideal Control*  $u_*(t)$  from:

$$\begin{cases} \dot{x}_* = A x_* + B u_* + \Gamma u_D; x_*(0) = x_0 \\ y_* = C x_* = y_m \end{cases} \quad (2.12)$$

Such ideal trajectories, if they exist, will be linear combinations of the reference model state and input, as shown in Equation (2.11), and will produce exact output tracking in a disturbance-free plant (2.12).

*Model Matching Conditions* are obtained by substituting (2.11) into (2.12) using (2.9) and (2.10):

$$\begin{aligned} AS_{11}^* + BS_{21}^* &= S_{11}^* A_m \\ AS_{12}^* + BS_{22}^* &= S_{12}^* F_m + S_{11}^* \\ AS_{13}^* + BS_{23}^* + \Gamma \theta &= S_{13}^* \\ CS_{11}^* &= C_m \end{aligned} \quad (2.13)$$

$$CS_{12}^* = 0$$

$$CS_{13}^* = 0$$

These are necessary and sufficient conditions for the existence of ideal trajectories in the form of Equation (2.11). Solutions to these matching conditions must exist for later analysis, but explicit solutions are not required for the adaptive controller design (Balas M. , 1995).

Theorem 1: If CB is nonsingular, there exist unique solutions to the Linear Matching Conditions (2.13) when no eigenvalues of  $A_m$ ,  $F_m$ , or  $F$  are transmission zeroes of  $A$ . See Section **Error! Reference source not found.** for proof (Frost, Balas, & Wright, 2009).

To achieve the desired control objective, i.e., for the plant to asymptotically track the output of the reference model, the output error vector is defined as

$$e_y \equiv y_p - y_m, \quad (2.14)$$

with the control goal:  $e_y \xrightarrow{t \rightarrow \infty} 0$ . Therefore, state tracking errors are defined as follows:

$$e_* \equiv x - x_* \quad (2.15)$$

Equations (2.12) and (2.15) are now used to develop the output error as:

$$e_y \equiv y - y_m = y - y_* = Cx - Cx_* = Ce_* \quad (2.16)$$

If we define  $\Delta u \equiv u - u_*$ , from (2.5) and (2.12) we have:

$$\dot{e}_* = Ae_* + B\Delta u \quad (2.17)$$

A fixed gain controller defined as

$$u_p = u_* + G_e^* e_y \quad (2.18)$$

can now be used in plant model (2.5) and combined with (2.12) and the output error from (2.16) to obtain the error dynamics tracking model:

$$\dot{e}_* = (A + BG_e^*C)e_* \quad (2.19)$$

The derivations above can be summarized as:

Theorem 2: If  $(A, B, C)$  is output feedback stabilizable with a gain  $G_e^*$ , (the eigenvalues of  $A_C \equiv A + BG_e^*C$  are all to the left of the  $j\omega$ -axis), then the fixed gain controller in (2.18) will produce local output tracking, i.e.,  $\lim_{t \rightarrow \infty} e_y = 0$  (NESC, 2016).

Note that output feedback stabilization can be accomplished when  $M + P + N_D > N$  and  $(A, B, C)$  is controllable and observable (Kimura, 1975). This does not require detailed knowledge of the parameter matrices, suggesting that an adaptive control architecture might be possible.

The control objective for the system of the plant (2.5) and disturbance generator (2.7) will be accomplished by an adaptive control law of the form:

$$u_p = G_m x_m + G_u u_m + G_e e_y + G_D \varphi_D \quad (2.20)$$

where  $G_m, G_u, G_e$  and  $G_D$  are gain matrices of compatible dimension. We make asymptotic output tracking possible by developing the gain adaptation laws by substituting  $z_D$  in the form given in (2.7) into (2.11) and using the result in (2.18):

$$\begin{cases} \Delta G_u \equiv G_u - S_{22}^* \\ \Delta G_m \equiv G_m - S_{21}^* \\ \Delta G_e \equiv G_e - G_e^* \\ \Delta G_D \equiv G_D - S_{23}^* L \end{cases} \quad (2.21)$$

*Ideal Trajectories* from (2.11), *Ideal Control* from (2.12), and the adaptive control law from (2.20), now allow us to define:

$$\begin{aligned} \Delta u &= u - u_* \\ &= \Delta G_u u_m + \Delta G_m x_m + (G_e^* + \Delta G_e) e_y + \Delta G_D \varphi_D \end{aligned} \quad (2.22)$$

We now expand (2.17) given (2.16) and (2.22), resulting in:

$$\begin{aligned}
\dot{e}_* &= Ae_* + B\Delta u \\
&= (A + BG_e^*C)e_* + B[\Delta G_u \quad \Delta G_m \quad \Delta G_e \quad \Delta G_D]\eta \\
&= A_c e_* + B\Delta G\eta
\end{aligned} \tag{2.23}$$

where  $\eta \equiv [u_m^T \ x_m^T \ e_y^T \ \varphi_D^T]$ . We now combine (2.16) and (2.23) and obtain the *Tracking Error*:

$$\begin{cases} \dot{e}_* = A_c e_* + B\Delta G\eta \\ e_y = C e_* \end{cases} \tag{2.24}$$

We now specify the *Adaptive Gain Laws*:

$$\dot{G} = -e_y \eta^T \sigma \tag{2.25}$$

where  $\sigma$  is the *gain weighting matrix* which is diagonal, positive definite (i.e.,  $\sigma \equiv \text{diag}[\sigma_u, \sigma_m, \sigma_e, \sigma_D] > 0$ ) used to tune the adaptation rate. This results in the following:

$$\begin{cases} \dot{G}_u = -e_y u_m^T \sigma_u \end{cases} \tag{2.26}$$

$$\begin{cases} \dot{G}_m = -e_y x_m^T \sigma_m \end{cases} \tag{2.27}$$

$$\begin{cases} \dot{G}_e = -e_y e_y^T \sigma_e \end{cases} \tag{2.28}$$

$$\begin{cases} \dot{G}_D = -e_y \varphi_D^T \sigma_D \end{cases} \tag{2.29}$$

### 2.3. Closed-loop Stability Analysis

The closed-loop adaptive system is now of the form

$$\begin{cases} \dot{e}_* = A_c e_* + B\Delta G\eta \\ e_y = C e_* \\ \Delta \dot{G} = \dot{G} = -e_y \eta^T \sigma \end{cases} \tag{2.30}$$

Because  $\Delta G \equiv G - G_*$  we are able to obtain (2.30) from (2.25) where  $G_* \equiv$

$[S_{22}^* \ S_{21}^* \ G_e^* \ S_{23}^*]$ . The stability of (2.30), which is a nonlinear system, can be analyzed

using Lyapunov Theory (Vidyasagar, 1993). In doing so, we first form the positive

definite functions:

$$\begin{cases} V_1(e) \equiv \frac{1}{2} e_*^T P e_* \\ V_2(\Delta G) \equiv \frac{1}{2} \text{tr}(\Delta G \sigma^{-1} \Delta G^T) \end{cases} \quad (2.31)$$

where  $P > 0$  is the solution of the following Kalman-Yacubovic Conditions:

$$\begin{cases} A_c^T P + P A_c \leq -\varepsilon I; \varepsilon > 0 \\ P B = C^T \end{cases} \quad (2.32)$$

From (Balas & Fuentes, 2004) we know existence of a symmetric positive definite solution of the Kalman-Yacubovic Conditions is known to be equivalent to the following:

$$T_c(s) \equiv C(sI - A_c)B \text{ strictly positive real (SPR)} \quad (2.33)$$

This means that, for some  $\varepsilon > 0$ ,

$$\text{Re } T_c(-\sigma + j\omega) \geq 0 \mid \forall \text{Re}(\omega) \quad (2.34)$$

This equivalence is proven in Appendix B of (Vidyasagar, 1993). Computing the derivatives  $\dot{V}_i$  along the trajectories of (2.31) using (2.32), we have

$$\begin{cases} \dot{V}_1 = -\frac{1}{2} e_*^T Q e_* + e_*^T P B \Delta G \eta + e_*^T P \Delta g = -\frac{1}{2} e_*^T Q e_* + e_y^T v + e_*^T P \Delta g \\ v \equiv \Delta G \eta \end{cases} \quad (2.35)$$

and

$$\begin{aligned} \dot{V}_2 &= \text{tr}(\Delta \dot{G} \sigma^{-1} \Delta G^T) = \text{tr}(-e_y \eta^T \sigma)(\sigma^{-1} \Delta G^T) \\ &= -\text{tr}(e_y v^T) = -\text{tr}(e_y^T v) = -e_y^T v \end{aligned} \quad (2.36)$$

We can now form

$$V \equiv V_1 + V_2 \Rightarrow \dot{V} = -\frac{1}{2} e_*^T Q e_* \quad (2.37)$$

with  $\dot{V} \leq 0$ . Therefore, stability of the zero equilibrium points of (2.31) are guaranteed and all trajectories of (2.32) remain bounded. This result, in turn, guarantees that all trajectories  $(e_*, \Delta G)$  are bounded. However, to prove convergence, i.e., that  $e_* \xrightarrow[t \rightarrow \infty]{} 0$

(and hence  $e_y \equiv y - y_m = Ce_* \xrightarrow[t \rightarrow \infty]{} 0$ ), we apply Barbalat's Lemma as described in

**Error! Reference source not found.** (NESC, 2016).

This stability analysis proves asymptotic tracking occurs with bounded adaptive gains. It does not prove convergence of  $\Delta G$  ( $\Delta G \xrightarrow[t \rightarrow \infty]{} 0$ ), although  $\Delta G$  convergence is not required for the adaptive controller to achieve its goals.

#### 2.4. Augmented Direct Adaptive Regulation and Disturbance Mitigation

Recall from Section 2.2 the feedback control law

$$u_p = G_m x_m + G_u u_m + G_e e_y + G_D \varphi_D$$

where the adaptive gain matrices  $G_m$ ,  $G_u$ ,  $G_e$ , and  $G_D$  are updated as follows:

$$\begin{cases} \dot{G}_u = -e_y u_m^T \sigma_u \\ \dot{G}_m = -e_y x_m^T \sigma_m \\ \dot{G}_e = -e_y e_y^T \sigma_e \\ \dot{G}_D = -e_y \varphi_D^T \sigma_D \end{cases}$$

Reference model terms are typically included in the control law to improve tracking performance (NESC, 2016). For application on Penny, the adaptive controller is meant to regulate excursions away from the operating region where the fixed gain controllers are designed. If it is assumed that  $y_m(t) = 0$ , the stability analysis above still holds, and the derived adaptive control law becomes (2.38), whose closed-loop system diagram is shown in Figure 2.1.

$$\begin{aligned} u_a &= G_e e_y + G_D \varphi_D \\ \begin{cases} \dot{G}_e &= -e_y e_y^T \sigma_e \\ \dot{G}_D &= -e_y \varphi_D^T \sigma_D \end{cases} \end{aligned} \tag{2.38}$$

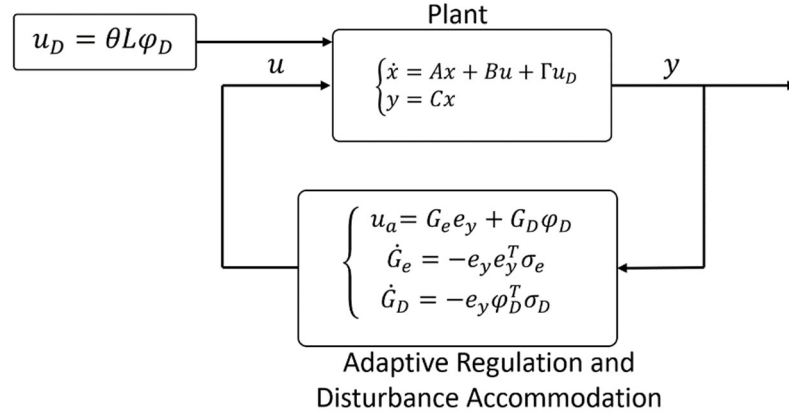


Figure 2.1 Direct adaptive state regulation and disturbance accommodation.

In (Balas & Frost, 2019) it was proven that any nonminimum phase LTI system controlled by a linear fixed-gain controller with stable actuator dynamics can be stabilized through the direct adaptive augmentation, shown in Figure 2.2.

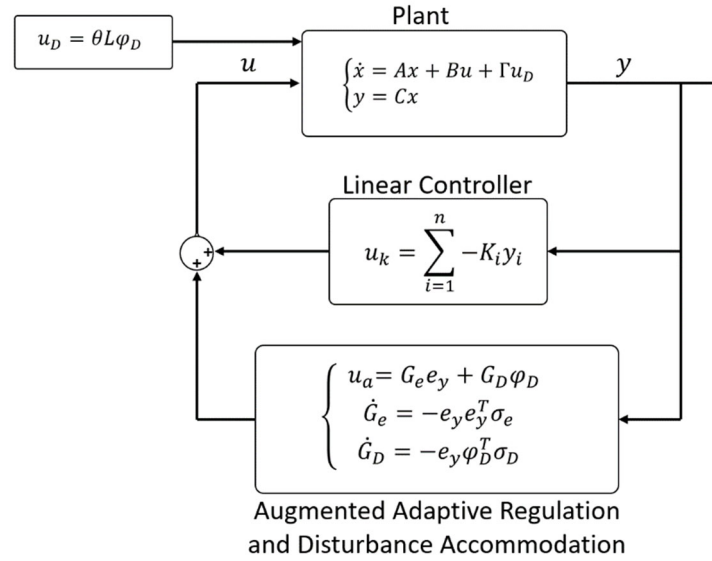


Figure 2.2 Direct adaptive state regulation and disturbance accommodation.

## 2.5. Sensor Blending

From Section 2.2 we know to that, guarantee asymptotic state convergence with bounded adaptive gains, the LTI system must be minimum phase (stable transmission

zeros) and have a positive real high frequency gain ( $CB > 0$ ). But Section 2.2 does not provide guidance on how to approach the adaptive augmentation of linear time-invariant systems that do not meet the ASPR criterion. Indeed, transmission zeros are invariant under coordinate transformations and linear feedback, so these approaches cannot be used to modify any unstable transmission zeros in an open-loop linear plant.

Here we use the output shaping algorithm presented in (Hartman, 2011), which allows us to arbitrarily place plant zeros using a linear combination of state outputs.

1. Nominal system is expressed in control canonical form, with *Controllability* matrix  $H$  and transformations:
  - a.  $A_{cc} = T^{-1}\bar{A}T$
  - b.  $B_{cc} = T^{-1}\bar{B}$
  - c.  $C_{cc} = CT$
  - d.  $T = H H^{-1}$
2. Determine desired numerator polynomial that meets ASPR requirements.
3. Define  $\tilde{C}_{cc} = [c_0 \ c_1 \ c_2 \ \dots \ c_{n-1}]$ , which are the coefficients of the desired numerator polynomial.
4. Transform  $\tilde{C}_{cc}$  into standard form:  $C_{cc} = \tilde{C}_{cc}T^{-1}$ , where  $\tilde{C}$  is the blended output.

This algorithm results in an output matrix that is a linear combination of the system state, as well as a new high-frequency gain  $\tilde{C}\bar{B} = 1$ . See Section 0 for Matlab Script.

### 3. Testbed Hardware

From the beginning of this research, high importance was placed on implementing the theoretical control concepts discussed herein on real hardware. Motivations which impacted hardware design were the desire to utilize parts on-hand to control costs, and that the system should be portable for ease of transport, setup, and demonstration. These early decisions doubtless led to the protracted development of the Inverted Pendulum Robot (“Penny”), since much effort was made to compensate for, and ultimately replace, sub-standard hardware.

The following sections detail the robotic testbed system, Penny, as it exists today. However, the development of Penny was an iterative process executed over several years as hardware testing and control software implementation campaigns identified deficiencies. Additional functionality was also added to Penny over time to increase capability and usability, and the rationale for these upgrades will be discussed in the following sections.

#### 3.1 Cart System

Penny is a cart-pole system with maximum cart dimensions shown in Figure 3.1. Significant components are further identified in Figure 3.2 and Figure 3.3.

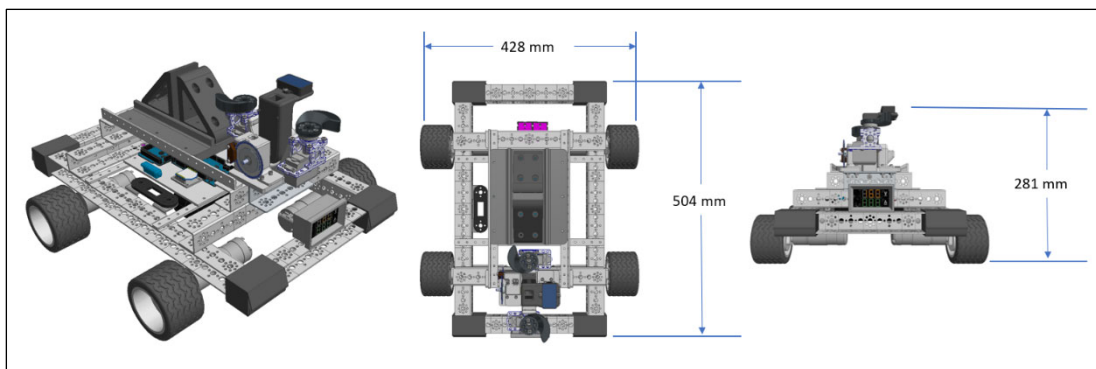


Figure 3.1 Penny cart CAD model: isometric view and maximum outer dimensions.

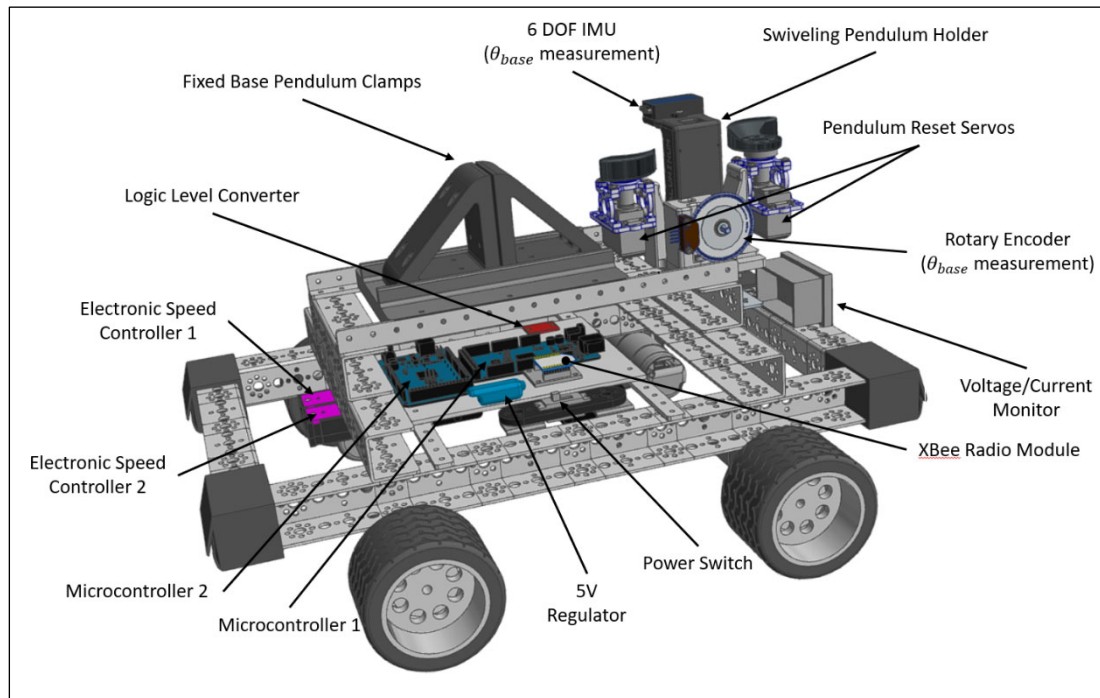


Figure 3.2 Upper Penny CAD model view with significant components identified.

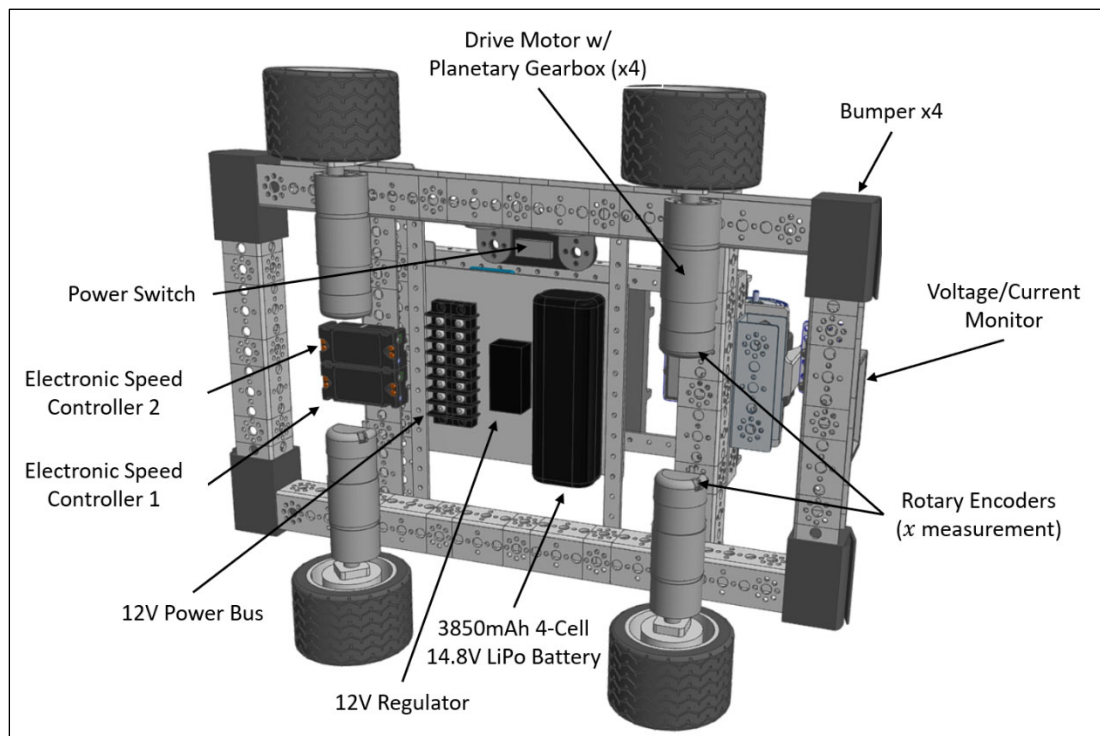


Figure 3.3 Lower Penny CAD model view with significant components identified.

The chassis of the cart system is comprised of aluminum extrusions, plates, and brackets manufactured by Pitsco™. These components were chosen for their availability and ease of assembly, since these parts come pre-drilled with standard mounting hole patterns. Four Actobotics 12V gear motors serve to actuate the cart, and rotary encoders built into the front two gear motors provide the wheel angular position measurements used to compute cart position ( $x$ ) and velocity ( $\dot{x}$ ).

Motor power is metered by two electronic speed controllers (ESC) with ESC1 powering the left side motors and ESC2 powering the right, giving Penny skid steering capability, although this was not used in this work. Power is fed to the ESCs via a 12V regulator. This was done when it was determined that one source of undesirable drive train dynamics stems from the variability in battery output voltage. A 4-cell lithium polymer (LiPo) battery provides electrical power to the 12V regulator. This battery fully charges to 16.8V and is nearly fully depleted around 13V, so the 12V regulator provides the ESCs with a steady 12V throughout the batteries' operational range. Early operations mandated the user check battery voltage often to protect the battery from discharging past recovery (by a “smart” charger) and to know when to remove it for recharging. This operational inefficiency was initially mitigated using a battery cell monitor and later improved through the addition of a total Voltage/Current monitor.

Table 3.1 Drive Train Hardware

Item	Description
Front Gear Motors	Actobotics 638326, 12V, 165 RPM, NE12 mag. encoders
Rear Gear Motors	RobotZone 638278, 12V, 165 RPM
Speed Controllers	VexPro Victor SP
Wheels	Traxxas 3674, Nylon, 2.2 inch
Tires	Pro-Line Striker II

### 3.2 Pendulum System

Figure 3.4 shows the hardware associated with utilizing the inverted pendulum system. The clamping structure on the left secures a long flat metal plate which represents one of the flexible structures available for analysis on Penny. This long metal plate (shown in Figure 3.5) is technically not a pendulum, as it does not have a single axis of rotation, yet it is briefly discussed here for convenience. Unlike a true inverted pendulum, this vertically cantilevered plate always has a base angle of zero. However, its tip angle, indicative of its flexed state, is measured.

On the right side of Figure 3.4 sits the swiveling pendulum system. The holder secures a rod of half-inch square cross section and is attached to the mounting structure via a steel axle and roller bearings, ensuring a low friction single-degree-of-rotation. Affixed to the axle is a rotary encoder which provides the pendulum's base angle measurement.

As various control laws were being implemented and tested on Penny, it soon became clear that a repeatable means to reset the pendulum to vertical would be beneficial. A set of servo-actuated cams was implemented to accomplish this task. These cams position the holder to a predetermined vertical position and rapidly rotate away, releasing the pendulum when the controller starts running.

It was also determined that the incremental encoder would always introduce some zero-angle error upon reset, since the reset cams would never perfectly zero the pendulum. The control system would then be commanded to balance the system at an angle that was not quite zero, and the cart would slowly accelerate in the direction of this error as the controller tried to hold a non-zero angle. To combat this issue, an IMU was

added to the pendulum base to provide absolute positioning with respect to the gravity vector.

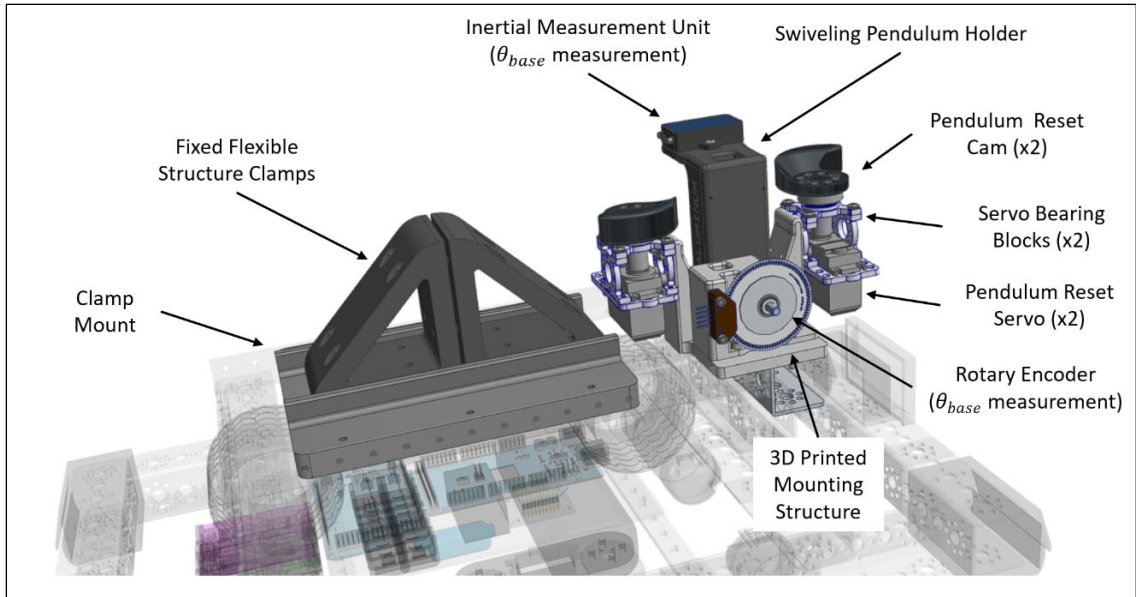


Figure 3.4 Penny pendulum system hardware on Cart.

### 3.3 Configurations

Figure 3.5 shows Penny's two principal configurations utilized under this work. The first configuration vertically cantilevers a flat aluminum plate. A gyroscope is mounted at the plate tip to report the angular rate of change (flex rate). The second configuration mounts a long square aluminum rod in Penny's swiveling pendulum holder. This rod is quite flexible and can also utilize a gyroscope at its tip for flex state information if desired.



Figure 3.5 Penny in (a) CFS and (b) FIP configurations

A brace can also be added to this flexible pendulum to work with the system's rigid body dynamics (although a small change in pendulum mass occurs as well). This brace is a thin aluminum L-beam extrusion that runs the exposed length of the pendulum, affixed at intervals with small patches of 3M™ Dual Lock™.

From this point forward, these configurations are abbreviated CFS and FIP, for Cantilevered Flexible Structure, and Flexible Inverted Pendulum, respectively.

### 3.4 Electrical Systems

The balance of Penny is comprised of electrical systems which store and distribute power, provide the user command and control capabilities, measure system states, and manage internal and external communications.

#### 3.4.1. Electrical Power Storage and Distribution

Electrical power is provided by a Lithium polymer (LiPo) battery. Any LiPo battery with 4-6 cells (14.8V to 22.2V nominal) that fits under the chassis can be used to power Penny, the lower bound set by the need to regulate down to the 12V motor voltage and the upper bound set by the maximum input voltage of the electronics regulator.

Battery power is first fed to the 5V electronics regulator which reduces battery voltage down to power the reset servos and the microcontrollers, which in turn power the robot's wired instrumentation (see Figure 3.6).

Battery power is then fed to the power switch which, when closed, sends power to the drive train's 12V regulator. The 12V regulator powers the two motor controllers, whose outputs are sent to the 12V power bus and distributed to the motors, one motor controller powering the two motors on each side of the robot.

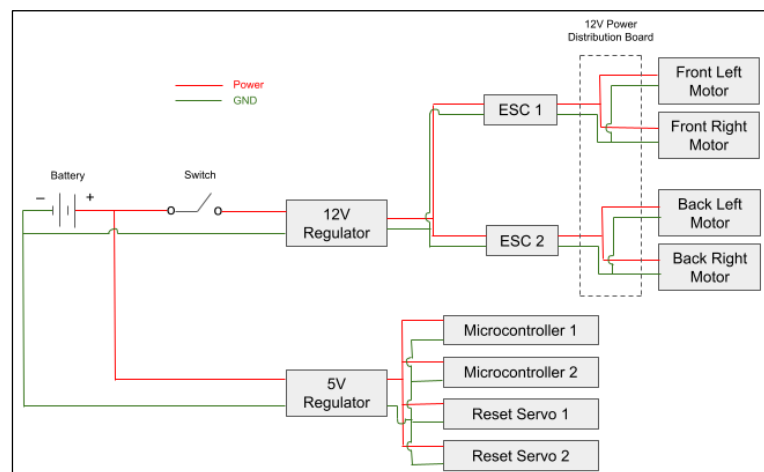


Figure 3.6 Penny Cart power distribution diagram.

Table 3.2 Power Storage and Distribution Main Components

Item	Description
Battery	ThunderPower G8 series 3850 mAh LiPo
5V Regulator	Castle Creations 10A BEC
12V Regulator	Castle Creations 20A BEC PRO

### 3.4.2. Command and Control

Command and control (C&C) functionality are provided by Microcontroller 1, an Arduino Mega Rev3. This board powers the three rotary encoders, receives their measurements, and communicates with the Xbee wireless module. It also builds the PWM signals that are sent to the ESCs, although under this work, the same signal is sent to both ESCs as the robot is intended to always drive straight when it serves as Beam or Pendulum actuation.

Microcontroller 1 also provides the main programming and telemetry output interface for the robot via its USB 2.0 port. This is where a laptop running Matlab/Simulink is connected. Simulink deploys a server program to Microcontroller 1 which executes Simulink programs on the Arduino, and which communicates telemetry back to Simulink for near real-time display and recording.

### 3.4.3. Instrumentation

As mentioned previously, Cart horizontal position ( $x$ ) and horizontal translation rate ( $\dot{x}$ ) are obtained by measuring the rotation of the front wheels. This is performed by magnetic rotary encoders mated to the front drive motors' rotor shaft and measured by Microcontroller 1. Penny's instrumentation power and data distribution systems are shown schematically in Figure 3.7 for the Cart and lower Pendulum, and in Figure 3.9 for the upper Pendulum.

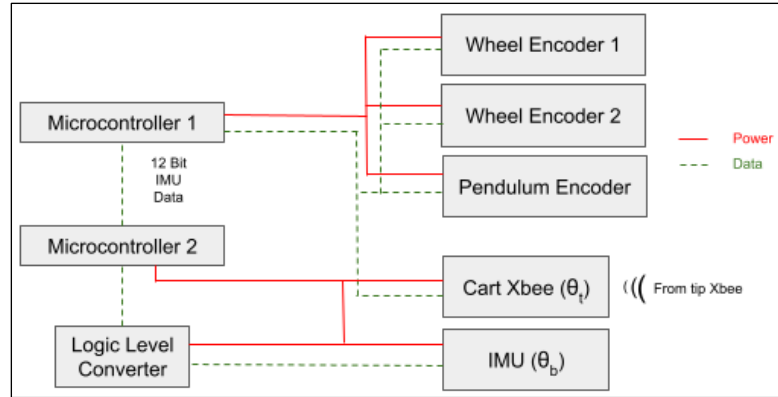


Figure 3.7 Cart instrumentation schematic.

The inverted pendulum base angle ( $\theta_b$ ) and base angular rate ( $\dot{\theta}_b$ ) are obtained using two instruments, a high-fidelity optical rotary encoder measuring angle relative to the starting angle, and an inertial measurement unit (IMU) measuring angle relative to the local gravity vector.

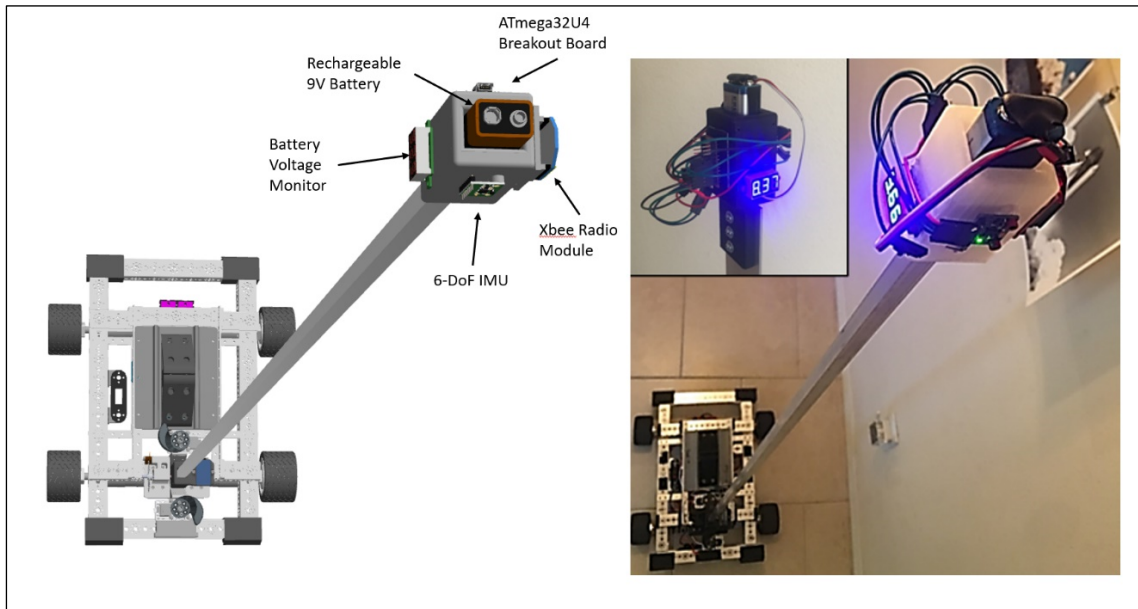


Figure 3.8 Tip instrumentation mounted on FIP and CFS (inset) configurations.

To extract explicit flex state information, an IMU was mounted to the top of the flexible structures in CFS and FIP configurations (reference Table 3.3 and Figure 3.8). As it was desirable not to route wires to this instrument and impact system dynamics, a wireless telemetry system was also implemented (reference Figure 3.8 and Figure 3.9).

Table 3.3 Instrumentation Components

Item	Description
Microcontroller 1,2	Arduino with ATmega 2650, 16 MHz microprocessor
Wheel Encoders	NE12 encoders, 2442.96 counts per wheel revolution
Pendulum Encoder	US Digital EM2 optical encoder, 0.036 degree resolution
Base IMU	MicroStrain® 3DM-6X3-25 AHRS
Radios, Cart and Tip	XB24-AWI-001 revE, 2.4 GHz, SparkFun Xbee Explorer
Tip Gyroscope	MPU 6050 MEMS IMU on GY-521 breakout board
Tip Microcontroller	Pro Micro with ATmega32U4, 16 MHz

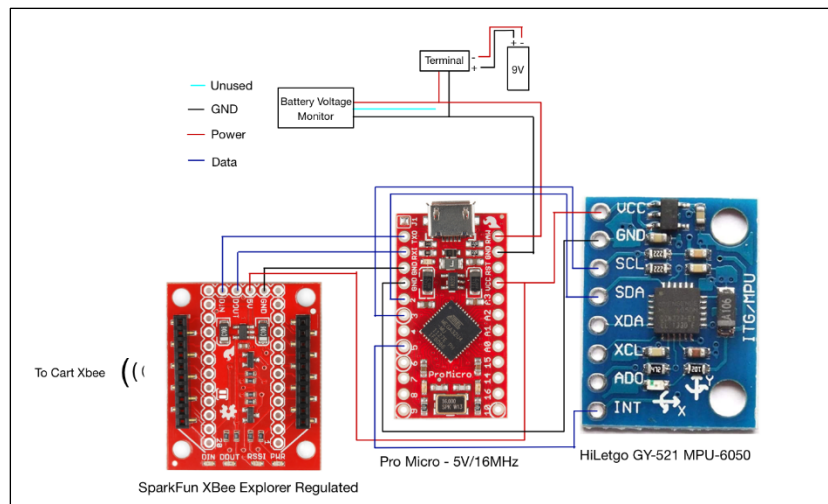


Figure 3.9 Tip instrumentation wiring schematic.

## 4. Testbed Software

The following section highlights the various software programs developed under this work, both internal to the robot and external from the Simulink user interface. Refer to Appendix 0 for more on the subject.

### 4.1. Software for Intra-robot Sensing and Communication

For the tip sensor measurements ( $\theta_t$  or  $\dot{\theta}_t$ , depending on IMU configuration), software was written in C using the Arduino Integrated Development Environment (IDE). This code reads MPU6050 gyroscope data for telemetering to Microcontroller 1 on the cart via a pair of Xbee radios (see Appendix 0 for code).

The code to read the base IMU sensor measurement ( $\theta_b$ ) on the swiveling pendulum holder was also written in C using the Arduino IDE. This code allows Microcontroller 2 to read the 3DM-6X3-25 IMU data via a logic-level shifter (see Appendix 0 for code). This program also builds the 12-bit DIO signal that sends the  $\theta_b$  measurement to Microcontroller 1 where it is subsequently made available to the Simulink user interface.

### 4.2. Penny Programming and Data Acquisition

Matlab's Simulink application is the main programming and data acquisition interface to Penny. In addition to Simulink's core functionality, two additional support packages were installed and used in data acquisition: Simulink Support Package for Arduino Hardware, and the Rensselaer Arduino Support Package. These support packages provide certain interface blocks allowing the hardware to be accessed through the Simulink UI. A Simulink model could then be deployed to the Arduino and run stand-alone while untethered from Simulink. In this case, Simulink models were run on the hardware in External Mode. This mode builds and deploys a server program to the target

Arduino, allowing near real-time communication between the Arduino and the Simulink UI.

Figure 4.1 shows the top-level Simulink command and control interface model for Penny FIP (the CFS interface is simpler and nearly identical). This model is launched from Matlab and all of the initialization parameters are written to the Matlab workspace by running a startup script. This program can be used to run control laws, command actuators, and acquire, view, and record data from the Penny robot.

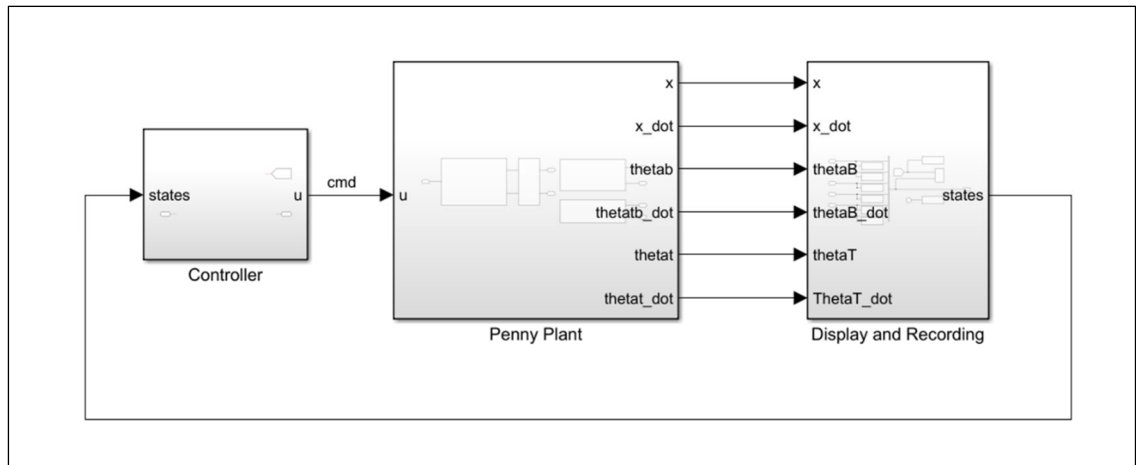


Figure 4.1 Simulink command and control programming interface for Penny FIP.

Unlike in simulation, no Penny plant model is required, as the Penny robot is the plant. Inside the Penny Plant subsystem, shown in Figure 4.2, are subsystems to command the actuators and receive wheel encoder data (left block), convert wheel data into cart position ( $x$ ) and velocity ( $\dot{x}$ ) states (middle block), measure pendulum base angle ( $\theta_b$ ) and base angular rate ( $\dot{\theta}_b$ ) states (upper right block), and measure pendulum tip angle ( $\theta_t$ ) and tip angular rate ( $\dot{\theta}_t$ ) states.

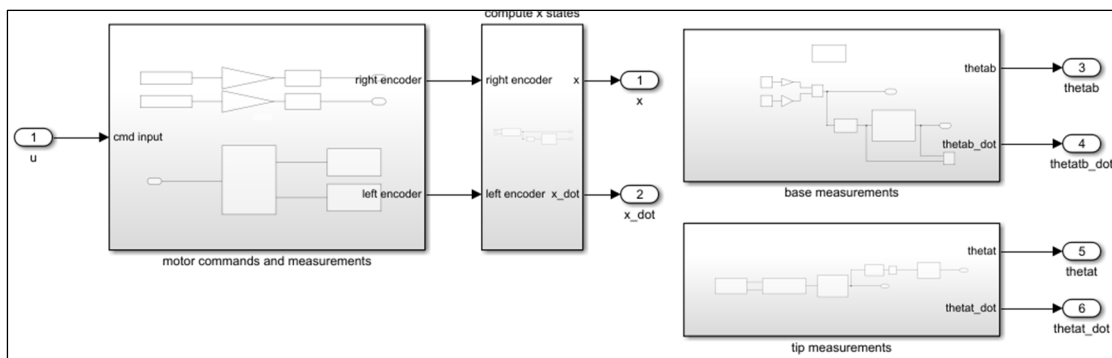


Figure 4.2 – Inside the Penny Plant subsystem block.

The upper left blocks highlighted in Figure 4.3 read the wheel encoders, providing raw measurements for cart position. The blocks in the lower right of Figure 4.3 send the actuation commands to the motors' electronic speed controllers. The middle subsystem block takes the command input and structures it for use by the ESCs. It also provides the user control over the deadband settings.

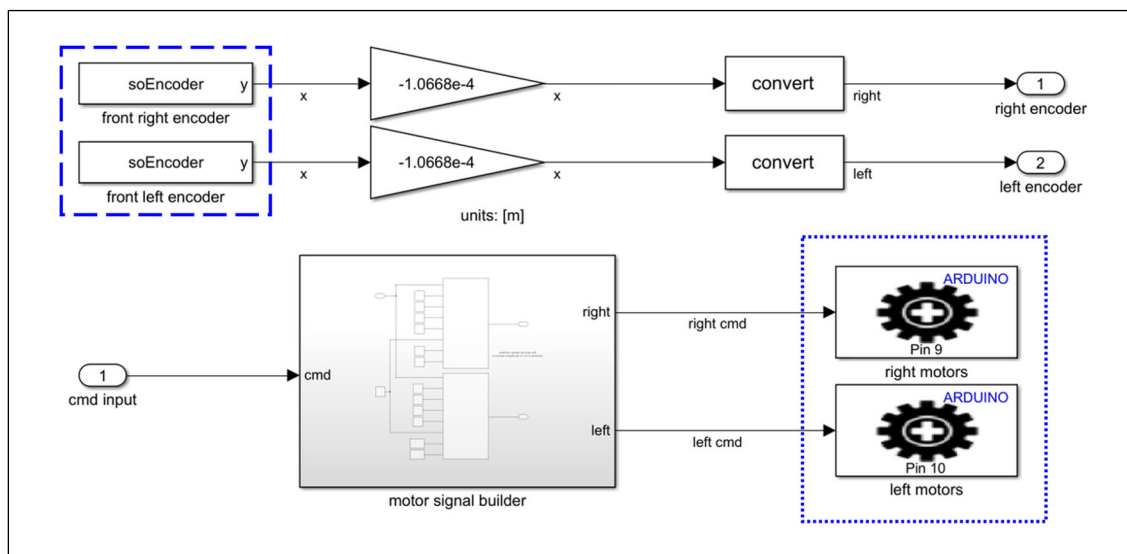


Figure 4.3 Motor commanding and wheel encoder reading subsystem.

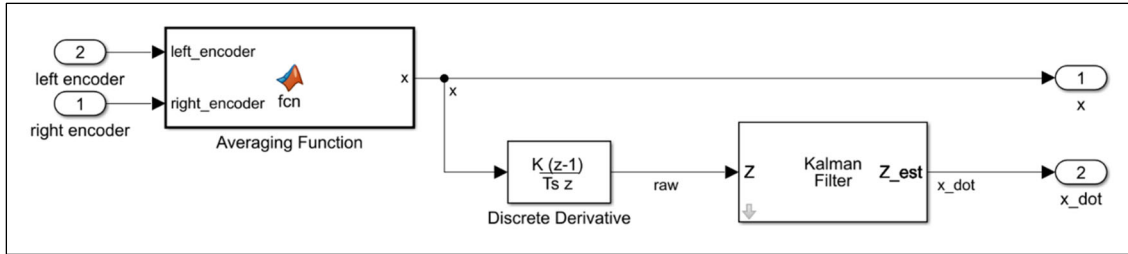


Figure 4.4 Acquisition of  $x$  and  $\dot{x}$  states.

The subsystem in Figure 4.4 computes  $x$  and  $\dot{x}$ . During system parameter identification testing, cart velocity is computed by filtering the derivative of cart position. Later, a state estimator was implemented to perform this function.

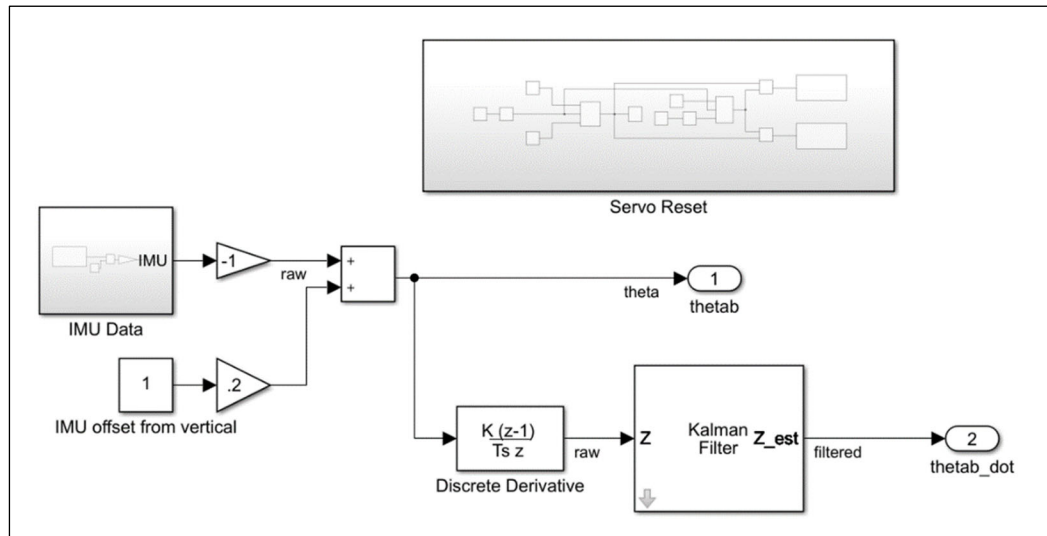


Figure 4.5 Acquisition of  $\theta_b$  and  $\dot{\theta}_b$  states from tip IMU.

The subsystem in Figure 4.5 computes pendulum base angle and angular rate ( $\theta_b$  and  $\dot{\theta}_b$ ) states. During system parameter identification testing,  $\dot{\theta}_b$  is computed by filtering  $\frac{d\theta}{dt}$ . Later, a state estimator was implemented to perform this function.

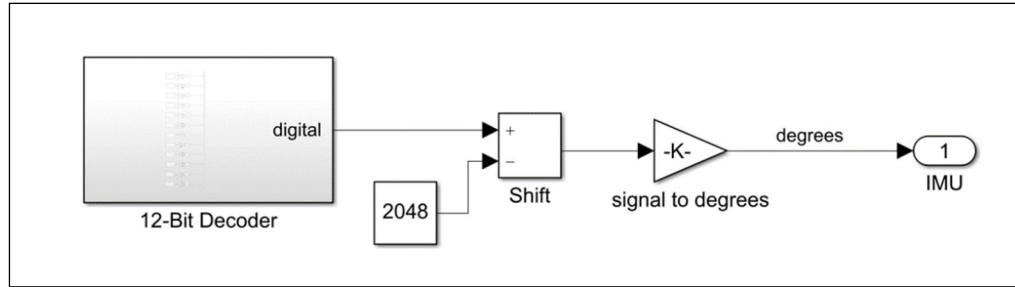


Figure 4.6 Decoding of the tip angle IMU's 12-bit signal.

Figure 4.6 shows the blocks used to decode the tip angle IMU's 12-bit signal. This signal is built on Microcontroller 2, which reads the pendulum base IMU, and is decoded on Microcontroller 1.

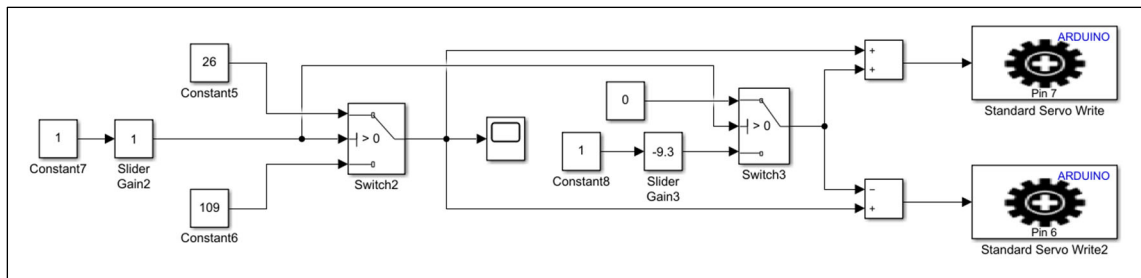


Figure 4.7 Command signal builder for pendulum reset servos.

The blocks in Figure 4.7 build the signal for pendulum reset servos. These servos clamp the inverted pendulum to vertical (or very nearly) on startup of Simulink code deployment to Microcontroller 1 and release the pendulum when the control software is enabled.

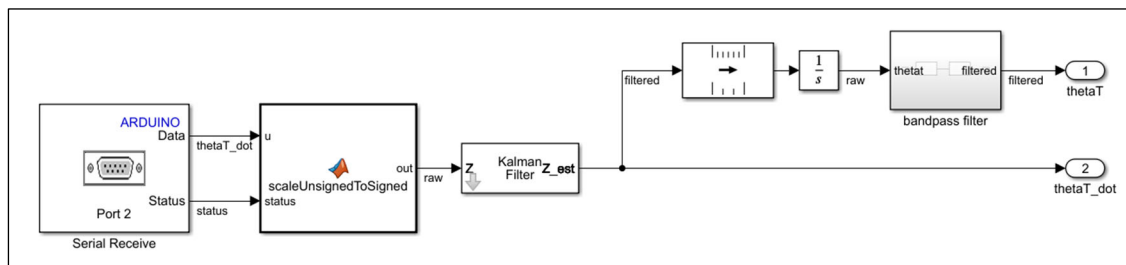


Figure 4.8 Acquisition of  $\theta_t$  and  $\dot{\theta}_t$  states.

Figure 4.8 shows the blocks that receive the tip gyro state data. Unlike the base IMU, which has excellent pre-installed filtering, the tip IMU's state data is always filtered in the Simulink control software.

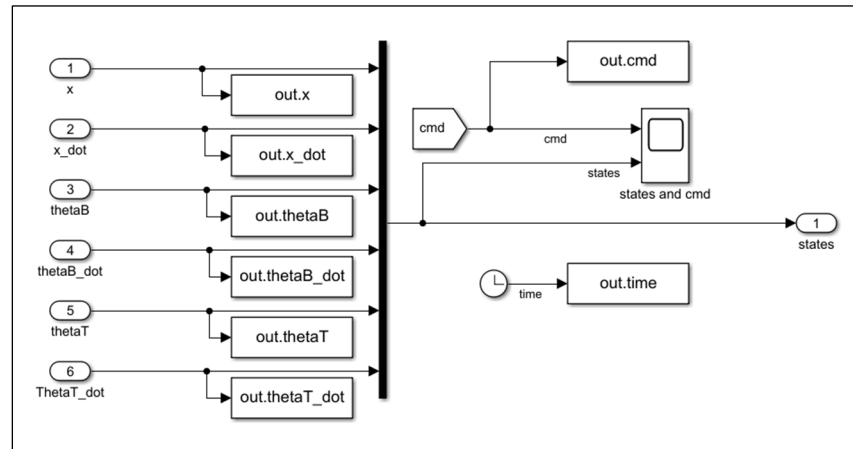


Figure 4.9 – State measuring and recording.

The state measuring and recording block from Figure 4.1 is expanded and shown in Figure 4.9. This block writes the state measurements to data files in the Matlab workspace and provides the scopes to visualize the data in near real-time.

## 5. Model Development

Before developing and applying adaptive augmentation to Penny, simulation models were developed for rapid deployment and testing of controller designs. These models needed to simulate Penny dynamics well enough such that controller performance in simulation would be reasonably representative of performance on the real hardware.

Also, recall from Section 2 that the control system architectures being applied to Penny guarantee state convergence only when certain criteria are met. The most fundamental of these criteria are that the system dynamics under analysis must be described as Linear Time-Invariant (LTI). After building and rigorously testing Penny, the author can state with confidence that Penny's dynamics are not linear. That being said, neither are the dynamics of any launch vehicle we apply linear control theory to, or probably any real system for that matter. The idea is that linear control theory holds if system dynamics can be reasonably modeled as linear around points of operation. To that end, a concerted effort was made to develop linear models of the Penny configurations utilized in this work.

However, if the linear models used for the Penny control system design are the same models the designed controllers run on, they will of course, function perfectly (and quite erroneously). It would also be instructive to be able to follow the same model development process in simulation as with the real robot. To that end, two model development campaigns were undertaken. The first would develop linear models for control theory applications requiring LTI systems. These would be generated using first principles, hardware parameters, and data collected from the operation of the systems with the idea that controllers developed using these LTI models would then be

implemented on hardware. The second would develop high-fidelity non-linear models to realistically emulate Penny dynamics even when operating far from equilibrium. These models would be used for rapid controller deployment and testing when testing on the robot was not feasible or practical. Controllers developed and tuned on the high-fidelity non-linear models would then be deployed to hardware, thereby streamlining the development process.

### 5.1. Oscillatory Dynamics Characterization

To determine natural frequencies ( $\omega_n$ ) and damping ratios ( $\zeta$ ) for the CFS and FIP configurations, the systems were excited, both by self-actuation and by the application of an external impulse, while angular state measurements were being recorded.

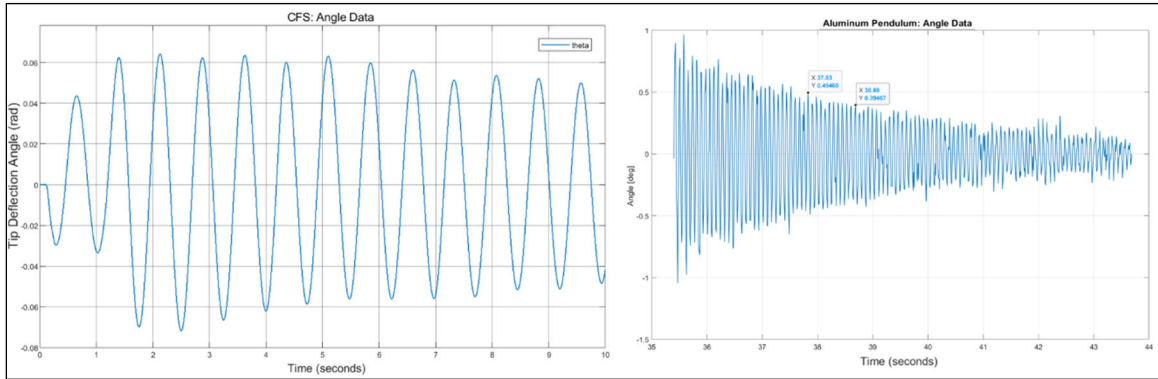


Figure 5.1 Recorded CFS (left) and FIP (right) angle data.

When freely oscillating, both systems are assumed to have principal vibration modes that behave as second-order harmonic oscillators of the Laplace domain form:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \quad (5.1)$$

where  $\omega_n$  is the system's natural frequency and  $\zeta$  its damping ratio. Natural frequency and damping ratio are now determined by deriving two equations from second order

modeling theory and two points from the flex angle data. Given the time and amplitude data  $(t_1, A_1)$  and  $(t_2, A_2)$ , the time difference and relative decay over the interval are,

$$\begin{cases} dt = t_2 - t_1 \\ dy = \frac{A_2}{A_1} < 1 \end{cases} \quad (5.2)$$

An exponential envelope of the form  $A(t) = A_0 e^{-\tau t}$  with time constant  $\tau = \omega_n \zeta$  is sought to fit the oscillatory response. Since

$$\begin{cases} A_1 = A(t_1) = A_0 e^{-\tau t_1} \\ A_2 = A(t_2) = A_0 e^{-\tau t_2} \end{cases} \quad (5.3)$$

then

$$\frac{A_2}{A_1} = dy = e^{-\tau(t_2 - t_1)} = e^{-\tau(dt)} \quad (5.4)$$

Therefore,

$$\ln(dy) = -\tau dt \rightarrow \tau = \frac{-\ln(dy)}{dt} \quad (5.5)$$

The initial value  $A_0$  is found by

$$A_0 = \frac{A_1}{e^{-\tau t_1}} = \frac{A_2}{e^{-\tau t_2}}. \quad (5.6)$$

The frequency ( $f$ ) in Hz of the relative decay is calculated from the number of cycles within the interval. The damping frequency is then  $\omega_d = 2\pi f$ .

For a second-order system, the natural frequency is related to the damping frequency by  $\omega_d = \omega_n \sqrt{1 - \zeta^2}$ . With known values  $\tau$  and  $\omega_d$ , the damping ratio and natural frequency are calculated by (5.7) and (5.8):

$$\zeta = \frac{1}{\sqrt{1 + \left(\frac{\omega_d}{\tau}\right)^2}} \quad (5.7)$$

$$\omega_n = \frac{\tau}{\zeta} \quad (5.8)$$

Table 5.1 Flexible structure oscillator parameters

	$\omega_n$ (rad/s)	$\zeta$
CFS	8.61	0.0040
FIP	71.6	0.0033

## 5.2. CFS Linear Model Development

To produce linear system models of Penny dynamics in CFS configuration, data sets were recorded by applying command step inputs ( $u$ ) to the system and recording the relevant state data. Figure 5.2 shows the Simulink model for CFS parameter identification.

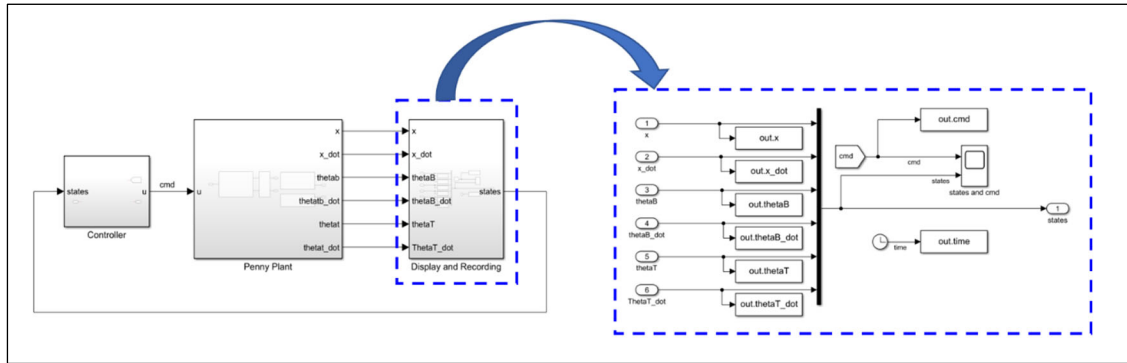


Figure 5.2 Simulink model for parameter ID data collection.

Next, the MATLAB system identification toolbox was used to identify state space model coefficients. These state space models were tested in Simulink by applying the same step inputs as used to produce the data.

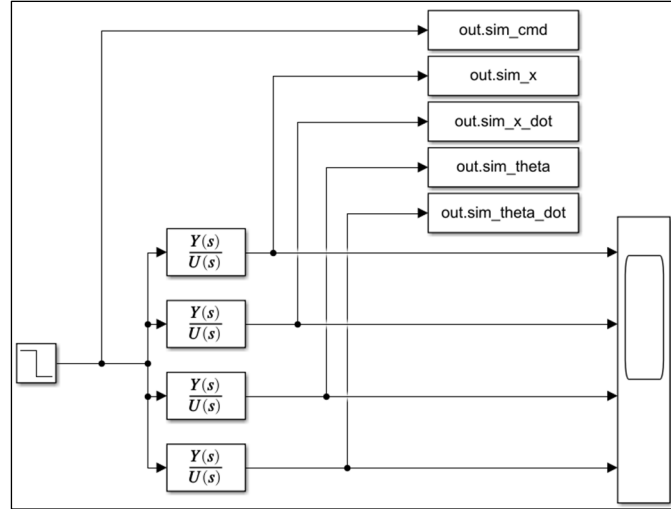


Figure 5.3 Simulink model to create linearized state data from transfer functions.

If the linearized state space models accurately reproduced state evolution near equilibrium as compared to the original non-linear data, the state space model would have been kept. As it was, errors were deemed unacceptably large, so the MATLAB system identification toolbox was used again to generate independent transfer functions for each input-output relationship. These transfer functions were then used to record linearized data given the same simulated step input (reference Figure 5.3) and the MATLAB system identification toolbox was again used to generate a state space model from the data.

The process described above resulted in the following linear model for Penny CFS cart states, as well as the first mode of oscillation:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -22.5 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 33.1 & -74.1 & -0.092 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0.2 \\ 0 \\ -0.34 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (5.9)$$

Figure 5.4 shows the output of the state space (linear) model and Penny CFS, given the same actuation command. Note the reasonably close agreement between models and also the linear system's unmodeled dynamics in the first second-or-so of the  $\theta$  and  $\dot{\theta}$  comparison plots. These higher-order dynamics rapidly attenuate and are assumed negligible for the time being. If it were determined that managing the system's first oscillatory mode was insufficient, higher order dynamics would be added and managed in similar fashion to those presented here. The otherwise close agreement between linear model and hardware suggests the linear model is a sufficiently good approximation of the physical system as to be useful henceforth for the application of linear systems analysis and control theory.

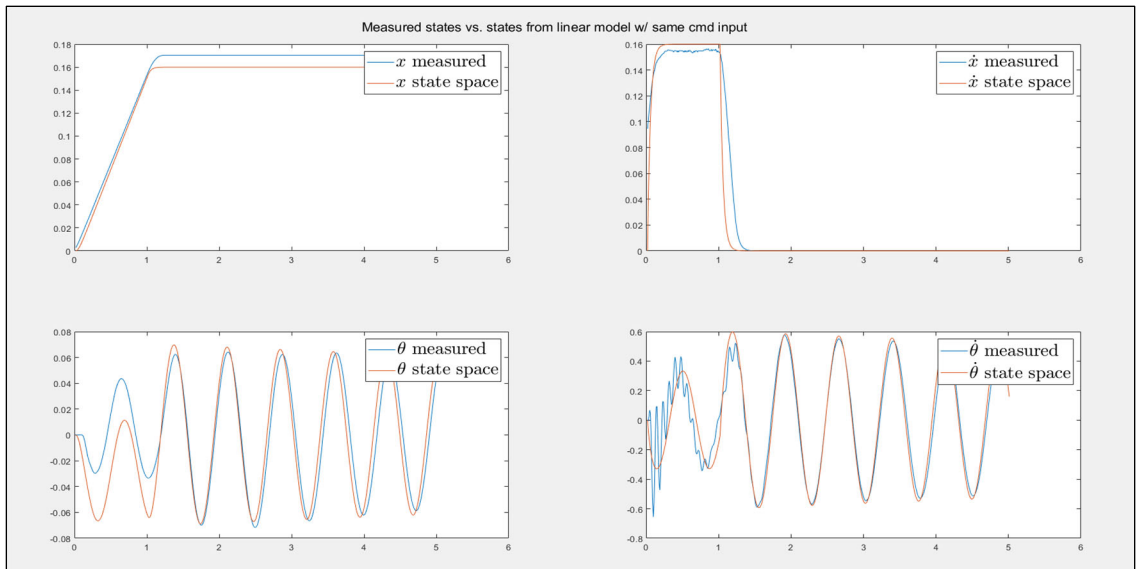


Figure 5.4 CFS Linear Model vs. Hardware for same input.

### 5.3. FIP Linear Model Development

To model a flexible inverted pendulum as a system of linear ordinary differential equations, a second-order harmonic oscillator representing pendulum flex dynamics was superimposed upon a rigid body dynamics model.

### 5.3.1. FIP Rigid Body Dynamics Modeling

The rigid body dynamics of the FIP system are given by the state space model from (5.16) where  $u_r = Fu$  is the force applied to the cart and  $x_r = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]^T$  are the system states defined in Table 5.2.

$$\dot{x}_r = A_r x_r + B_r u_r \quad (5.10)$$

Table 5.2 Summary of State Variables

State Variable	Description	Unit
$x$	Position of the cart	$m$
$\dot{x}$	Position rate of the cart	$m/s$
$\theta$	Angle of the pendulum	$rad$
$\dot{\theta}$	Angle rate of the pendulum	$rad/s$

The matrices  $A_r$  and  $B_r$  from (Florian, 2005) are defined as:

$$A_r = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-b(J+ml^2)}{J(M+m)+(Mml^2)} & -m^2 l^2 g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{bml}{J(M+m)+(Mml^2)} & \frac{mgl(M+m)}{J(M+m)+(Mml^2)} & 0 \end{pmatrix}, \quad (5.11)$$

$$B_r = \begin{pmatrix} 0 \\ \frac{J+ml^2}{J(M+m)+(Mml^2)} \\ 0 \\ \frac{-ml}{J(M+m)+(Mml^2)} \end{pmatrix},$$

where  $J = \frac{1}{12}m(2l)^2$  is the moment of inertia of the rod about its center of gravity. The parameters specific to Penny FIP are summarized in Table 5.3. The resulting FIP rigid body state space model is shown in (5.12).

Table 5.3 Rigid body parameters of Penny FIP

Parameter	Description	Value	Unit
$M$	Mass of the cart	5.35	$kg$
$m$	Mass of the rod	0.8066	$kg$
$l$	Length of rod from geometric center	0.917	$m$
$b$	Coefficient of friction	48.2	$N$
$g$	Gravitational acceleration	9.81	$m/s^2$
$J$	Moment of inertia	0.2444	$kg \cdot m^2$

$$\begin{aligned}
 A_r &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -8.682 & -5.803 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 6.829 & 8.557 & 0 \end{bmatrix}, \quad B_r = \begin{bmatrix} 0 \\ 0.1831 \\ 0 \\ -0.252 \end{bmatrix}, \\
 C_r &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad D_r = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},
 \end{aligned} \tag{5.12}$$

In the linearized cart-pole dynamics model above, recall that matrix  $B_r$  operates on input  $u_r$ , which has units of force. Penny FIP produces this force using gearmotor actuators, which need to be modeled as well.

### 5.3.2. Actuator Dynamics Model

The dynamics of a DC motor can be described by the state space model in (5.13) where  $u_a = V_{in}$  is the input voltage and  $x_a = [I \ \omega]^T$  are the system states defined in Table 5.4.

$$\dot{x}_a = A_a x_a + B_a u_a \tag{5.13}$$

Table 5.4 Summary of Actuator State Variables

State Variable	Description	Unit
$I$	Motor current	$A$
$\omega$	Motor angular speed	$rads$

The state space matrices  $A_a$  and  $B_a$  are defined as:

$$A_a = \begin{bmatrix} \frac{-R_m}{L_m} & \frac{-K_m}{L_m} \\ \frac{K_m}{J_{eq}} & \frac{-b_r}{J_{eq}} \end{bmatrix}, B_a = \begin{bmatrix} \frac{1}{L_m} \\ 0 \end{bmatrix}, \quad (5.14)$$

with the parameters as shown in Table 5.5. These parameters were determined by matching the model response to the experimental data given the same command input using Matlab's parameter identification application. These experimental data were recorded with Penny FIP operating on a flat surface with a motor step command of 20. No wheel slip was observed during the experiment or in the recorded data.

Since all four motors are driven by the same command, and therefore rotate with the same (approximately) magnitude and direction, the actuator model considers the sum of all wheel torques.

Table 5.5 Actuator Parameters

Parameter	Description	Value	Unit
$R_m$	Motor armature resistance	3.515e-05	$\Omega$
$K_m$	Motor torque constant	0.01090	$N \cdot m/A$
$L_m$	Motor armature inductance	0.01758	$H$
$J_{eq}$	Total rotor moment of inertia	3.386e-07	$kg \cdot m^2$
$b_r$	Viscous friction coefficient	0.001398	$N \cdot m \cdot s/rad$
$K_G$	Gearhead ratio	50.895	-
$r$	Wheel radius	0.043	$m$

Given these parameters, the state space expression for the actuator dynamics is shown by (5.15). Actuator functions mapping applied voltage to current draw and rotor angular velocity are provided in (5.16) and (5.17). The actuator model's transfer function poles are listed in (5.18), indicating stable actuator dynamics.

$$A_a = \begin{bmatrix} -0.002 & -0.62 \\ 3.2192e4 & -4.128e3 \end{bmatrix}, B_a = \begin{bmatrix} 56.88 \\ 0 \end{bmatrix}, \quad (5.15)$$

$$C_a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D_a = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$TF_{V \rightarrow I} = \frac{56.88s + 2.348e5}{s^2 + 4128s + 1.997e4} \quad (5.16)$$

$$TF_{V \rightarrow \omega} = \frac{1.8312e5}{s^2 + 4128s + 1.997e4} \quad (5.17)$$

$$Poles = \begin{bmatrix} -4.8427 \\ -4122.9 \end{bmatrix} \quad (5.18)$$

The torque applied to the wheel and the wheel angular speed are related to the motor torque and speed by the gearbox ratio  $K_G$ .

$$\tau_w = K_G \tau \quad (5.19)$$

$$\omega_w = \frac{\omega}{K_G}$$

The motor angular speed  $\omega$  is given by  $\omega = \frac{1}{K_m} V_{in} - \frac{R_m}{K_m^2} \tau$ . Rearranging terms yields

$\tau = \frac{K_m}{R_m} V_{in} - \frac{K_m^2}{R_m} \omega$ . The torque applied to the wheel is also equal to  $\tau_w = F_w r$  where  $r$  is the wheel radius. Therefore,

$$K_G \tau = F_w r \rightarrow F_w = K_G \left( \frac{K_m}{r R_m} V_{in} - \frac{K_m^2}{r R_m} \omega \right) \quad (5.20)$$

The control input  $u_r$  is now defined as  $F_w$ :

$$u_r = F_w = \left( \frac{K_m K_G}{r R_m} \right) V_{in} - \left( \frac{K_m^2 K_G}{r R_m} \right) \omega \quad (5.21)$$

Setting  $p_1 = \left( \frac{K_m K_G}{r R_m} \right)$  and  $p_2 = \left( \frac{K_m^2 K_G}{r R_m} \right)$  yields  $u_r = p_1 V_{in} - p_2 \omega$ . Finally, commands and input voltage are related by

$$G = \begin{cases} 0.1349\sigma, & \sigma \geq 0 \\ 0.1431\sigma, & \sigma < 0 \end{cases} \quad (5.22)$$

where  $\sigma$  is bounded such that  $\sigma \in \{-79, 79\}$  due to hardware constraints.

### 5.3.3. Combined Rigid Body and Actuator Dynamics

The relationship  $u_r = p_1 V_{in} - p_2 \omega$  is now substituted into the rigid body model

(5.10) as follows:

$$\begin{aligned} \dot{x}_r &= A_r x_r + B_r (p_1 V_{in} - p_2 \omega) \\ u &\triangleq V_{in} \end{aligned} \quad (5.23)$$

with the control input  $u = V_{in}$ , the state space matrices of the combined system are rewritten in the form  $\dot{x}_p = A_p x_p + B_p u$ , where  $x_p = [x_r \ x_a]^T$ . The block diagram shown in Figure 5.5 illustrates the interaction between control input and plant dynamics, where operator  $G$  maps the control signal to the actuator's input voltage.

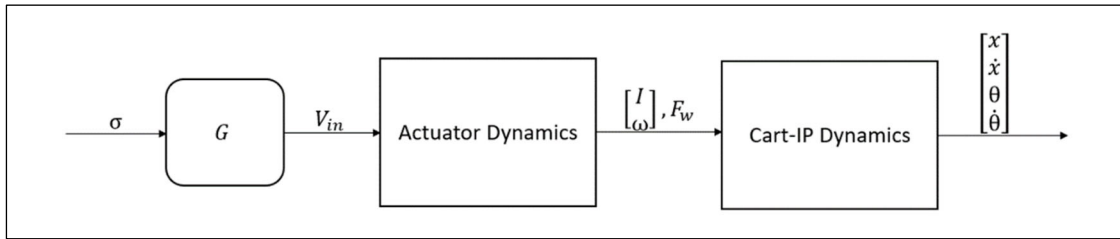


Figure 5.5 Block diagram of interaction between actuator and plant dynamics.

After many simulations and empirical recording runs, a scalar correction factor  $k_c = 4.69e4$  was added to the modified  $u_r$  input to numerically fit the state responses of the model to the experimental data. This resulted in

$$u_r = \frac{1}{k_c} (p_1 V_{in} - p_2 \omega) = 7.8182 V_{in} - 0.0852 \omega, \quad (5.24)$$

yielding the combined system model:

$$\underbrace{\begin{bmatrix} \dot{x}_r \\ \dot{x}_a \end{bmatrix}}_{\dot{x}_p} = \underbrace{\begin{bmatrix} A_r & L \\ 0_{2 \times 4} & A_a \end{bmatrix}}_{A_p} \underbrace{\begin{bmatrix} x_r \\ x_a \end{bmatrix}}_{x_p} + \underbrace{\begin{bmatrix} B_r' \\ B_a \end{bmatrix}}_{B_p} \underbrace{V_{in}}_u, \text{ where} \quad (5.25)$$

$$L = \begin{bmatrix} 0 & 0 \\ 0 & -\frac{p_2}{k_c} B_r(2,1) \\ 0 & 0 \\ 0 & -\frac{p_2}{k_c} B_r(4,1) \end{bmatrix}, B_r' = \begin{bmatrix} 0 \\ \frac{p_1}{k_c} B_r(2,1) \\ 0 \\ \frac{p_1}{k_c} B_r(4,1) \end{bmatrix}, \quad (5.26)$$

Actuator and plant dynamics are now described by a single state space model as shown schematically in Figure 5.6.

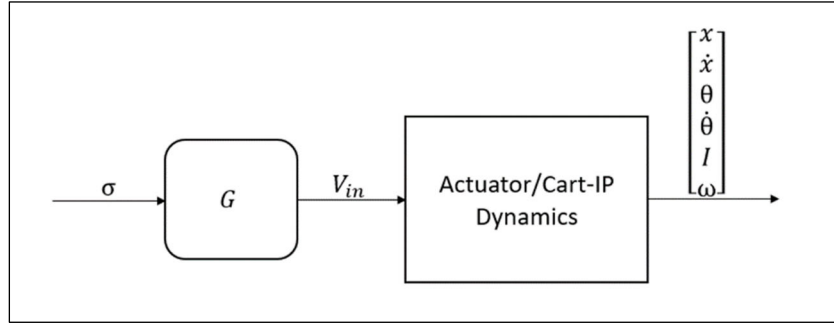


Figure 5.6 FIP Actuator and Rigid Body combined dynamics

Figure 5.7 and Figure 5.8 compare hardware and simulation states produced by sending each the same command signal,  $u = 20$  and  $u = 10$  respectively.

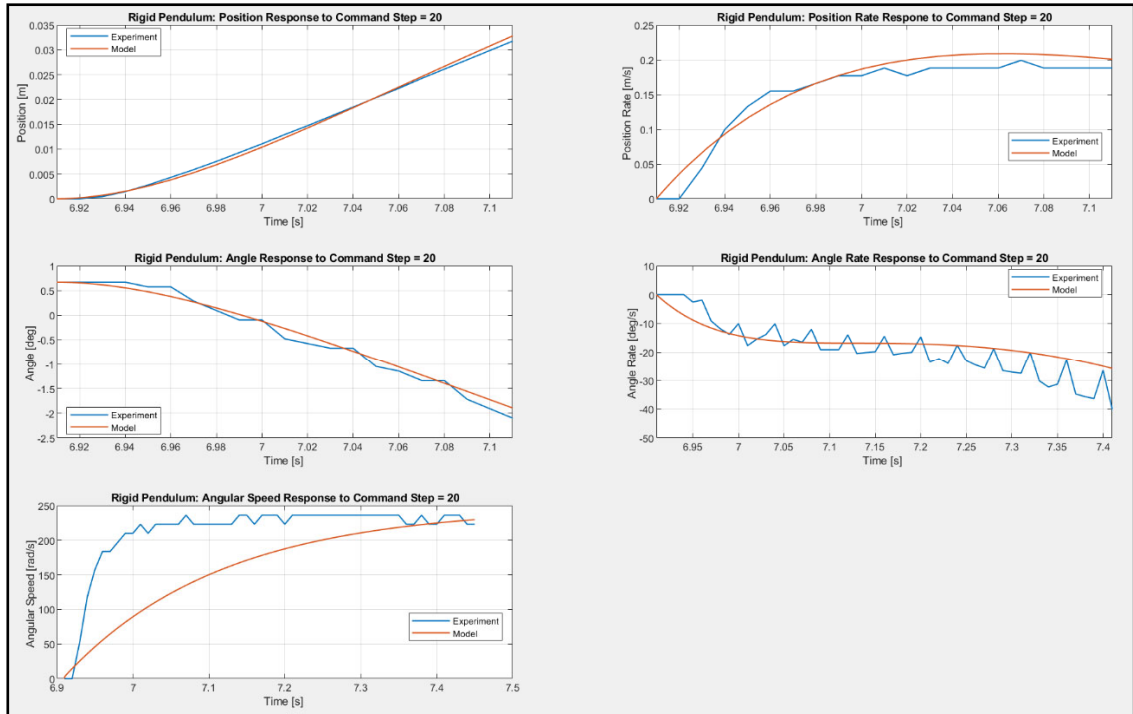


Figure 5.7 Linear Model vs. Hardware Data for Same Input Command,  $u = 20$

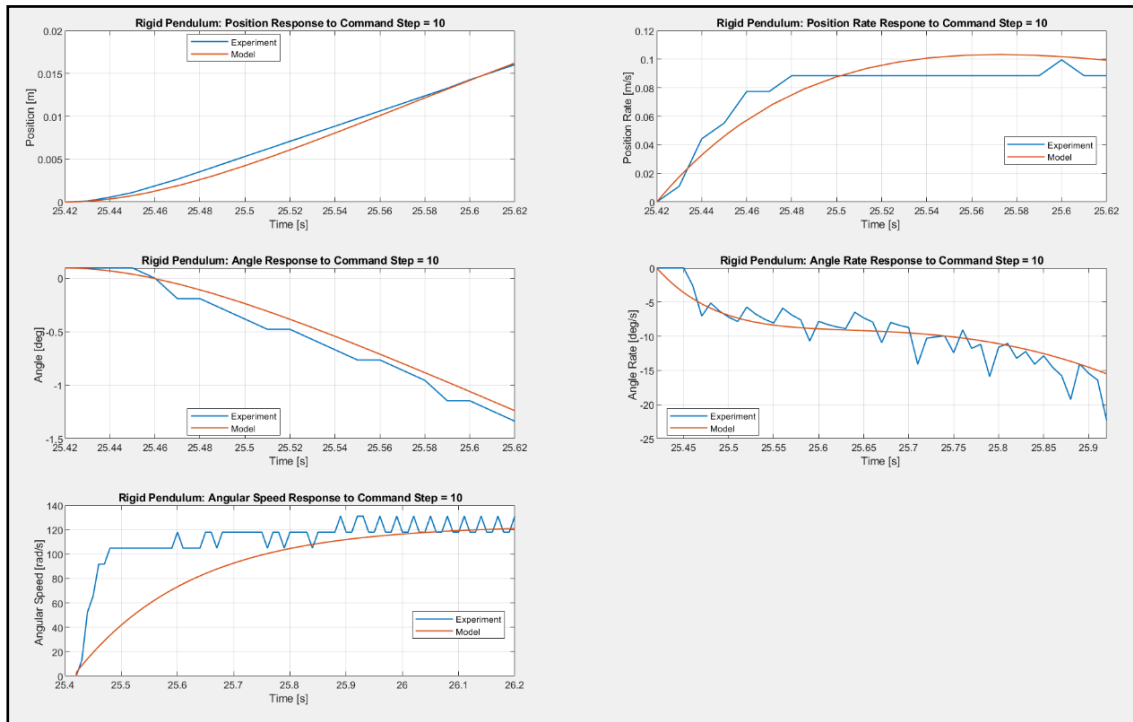


Figure 5.8 Linear Model vs. Hardware Data for Same Input Command,  $u = 10$

### 5.3.4. FIP Flex Dynamics Modeling

To incorporate FIP pendulum flexible dynamics with the rigid body dynamics previously modeled, two separate approaches were undertaken. This is because the FIP configuration initially had no tip sensor ( $\dot{\theta}_t$ ), only knowledge of the base angle states via  $\theta_b$ . The approach taken in this case was to superimpose second-order oscillator dynamics, i.e., a mass-spring-damper system, onto the rigid body model developed previously, as suggested in (Orr J. , 2011). When the tip sensor was added later, a new flex state was defined as the difference between base and tip angles (or angular rates, since the suppression of this state also guarantees angle agreement in a beam that has not been plastically deformed).

For the first flex modeling case, a mass-spring-damper system with virtual mass  $m_s$ , spring constant  $k$ , and damping constant  $c$  is modelled as acting at the center of the mass of the pendulum and stretching with respect to its own center of mass  $S$ , as shown in Figure 5.9.

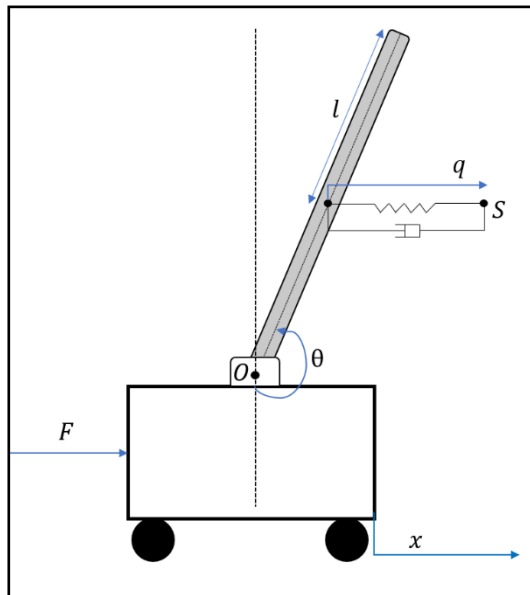


Figure 5.9 Schematic for flexible inverted pendulum model.

The differential equation of motion for a mass-spring-damper system is given by a second-order oscillatory model with the forcing function  $U_{osc}$ :

$$\ddot{q} + 2\zeta\omega_n\dot{q} + \omega_n^2 q = \frac{U_{osc}}{m_s}, \quad (5.27)$$

$$U_{osc} = J\ddot{\theta}$$

where  $J$  is the moment of inertia of the pendulum and  $m_s$  is the virtual mass of the mass-spring-damper system.

This results in a Laplace domain system:

$$s^2 Q(s) + 2\zeta\omega_n s Q(s) + \omega_n^2 Q(s) = \left(\frac{J}{m_s}\right) s^2 \theta(s) \quad (5.28)$$

The transfer function  $G(s)$  from angle  $\theta$  to the deflection state  $q$  is then given by:

$$G(s) = \frac{Q(s)}{\theta(s)} = \frac{(J/m_s)s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (5.29)$$

where  $\omega_n = 71.6 \text{ rad/s}$  and  $\zeta = .00327$  for the long aluminum inverted pendulum. The interaction between the rigid and flexible models is now shown in Figure 5.10. The rigid body angle  $\theta_r$  is the output of the rigid body model and it acts as the forcing function for the flexible dynamics model producing the deflection state  $q$ . These quantities are summed to produce an output angle that is a function of the states  $\theta_r, q, \dot{q}$ .

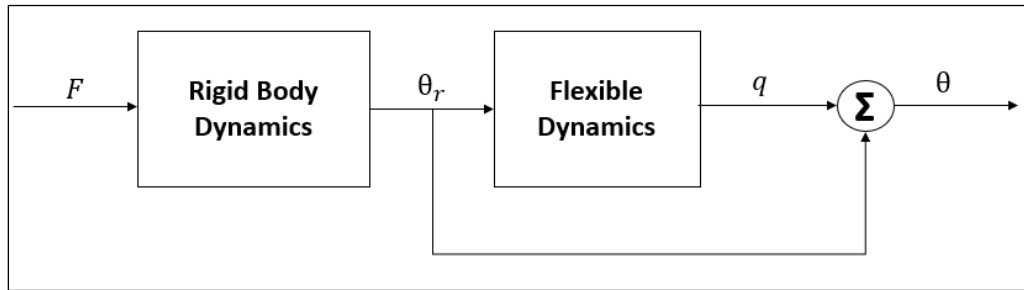


Figure 5.10 Rigid body and flexible dynamics model

To obtain a combined model of the rigid body and flexible dynamics, the system shown in Figure 5.10 is implemented in Simulink as shown in Figure 5.11.

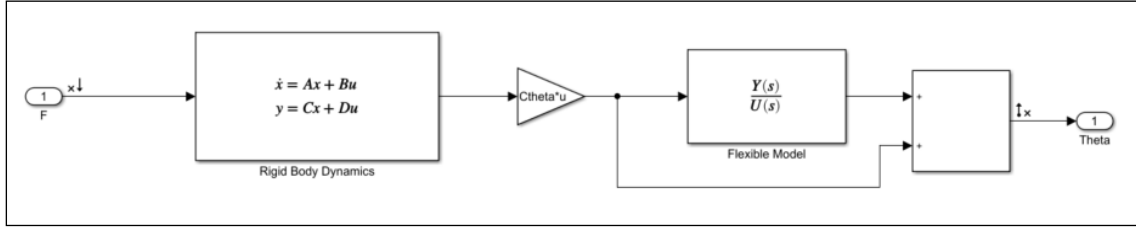


Figure 5.11 – Simulink model for linearized system.

The Matlab functions *linmod* or *linearize* are now employed on the Simulink model to generate the state space model representing the dynamics from the input  $F$  to output  $\theta$ .

The state vector for the described system is now  $x = [x \quad \dot{x} \quad \theta \quad \dot{\theta} \quad q \quad \dot{q}]^T$  and the control input is  $u = F$ , the force acting on the cart. The state space representation  $\dot{x} = A_{rf}x + B_{rf}u$ , which combines the rigid body and flexible dynamics is now:

$$A_{rf} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -8.872 & -0.9775 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 6.986 & 8.078 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -5131 & -0.4685 \end{bmatrix}, B_{rf} = \begin{bmatrix} 0 \\ 0.1841 \\ 0 \\ -0.1449 \\ 0 \\ 0 \end{bmatrix} \quad (5.30)$$

$$C_{rf} = [0 \quad 0 \quad 1.1 \quad 0 \quad -513.1 \quad -0.0469], \quad D_{rf} = [0]$$

The first four eigenvalues of matrix  $A_{rf}$ , summarized in (5.25), are associated with Penny FIP's linearized rigid body dynamics. The final two eigenvalues are associated with the system's linearized flexible dynamics.

$$\begin{array}{c} 0 \\ -8.9664 \\ 2.7367 \\ -2.6423 \\ -0.2343 + 71.6306i \\ -0.2343 - 71.6306i \end{array} \quad (5.31)$$

This revised model summarized by (5.25) is now added to the actuator model described in Section 5.3.2 to complete the open-loop system of the *Penny* robot in FIP configuration, shown in the Simulink implementation in Figure 5.12.

$$\begin{aligned} \dot{x} &= A_{rf}x + B_{rf}u \\ y &= C_{rf}x + D_{rf}u \end{aligned} \quad (5.32)$$

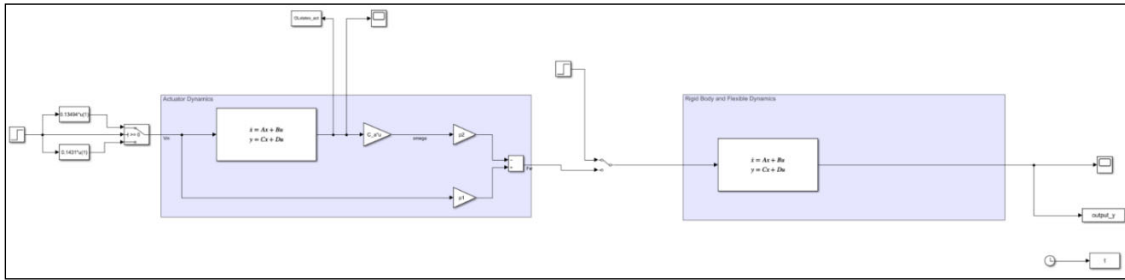


Figure 5.12 Model for combined system dynamics

Figure 5.13 shows the base angle response to a disturbance. At the time the FIP system was not being actively balanced under closed-loop control, so no vehicle state data were available for comparison. The dynamics of the flex states ( $q, \dot{q}$ ) feeding back into the base angle state ( $\theta_b$ ) appeared reasonable as frequency was a close match to prior excitation data without control. This model is now used for testing controllers in simulation before deployment to the Penny robot.

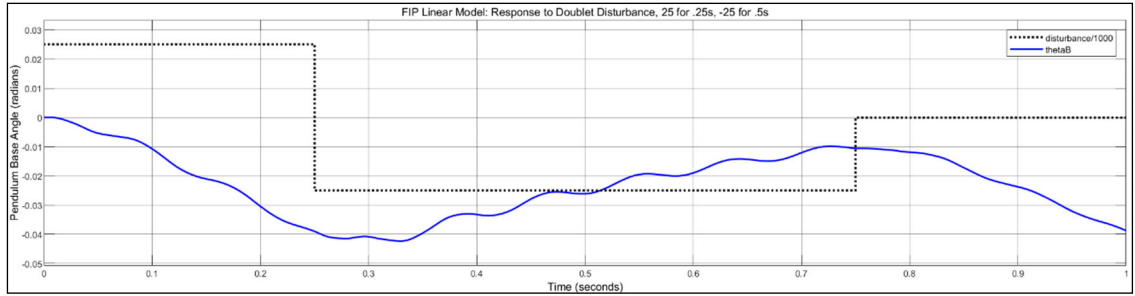


Figure 5.13 FIP combined linear model response to doublet disturbance.

### 5.3.5. FIP Disturbance State Modeling

Although essential for Penny CFS, a sensor reporting the state of FIP's pendulum tip was also deemed useful for flex state estimation. A 3D printed end cap was designed to accommodate the FIP inverted pendulum, and the hardware was transferred to this new part and secured to FIP.

Figure 5.14 shows angle state measurements from two consecutive FIP “pluck” tests, where the inverted pendulum tip is held while the center is manually deflected. Both tip and center are simultaneously released, causing the system to oscillate as it falls over.

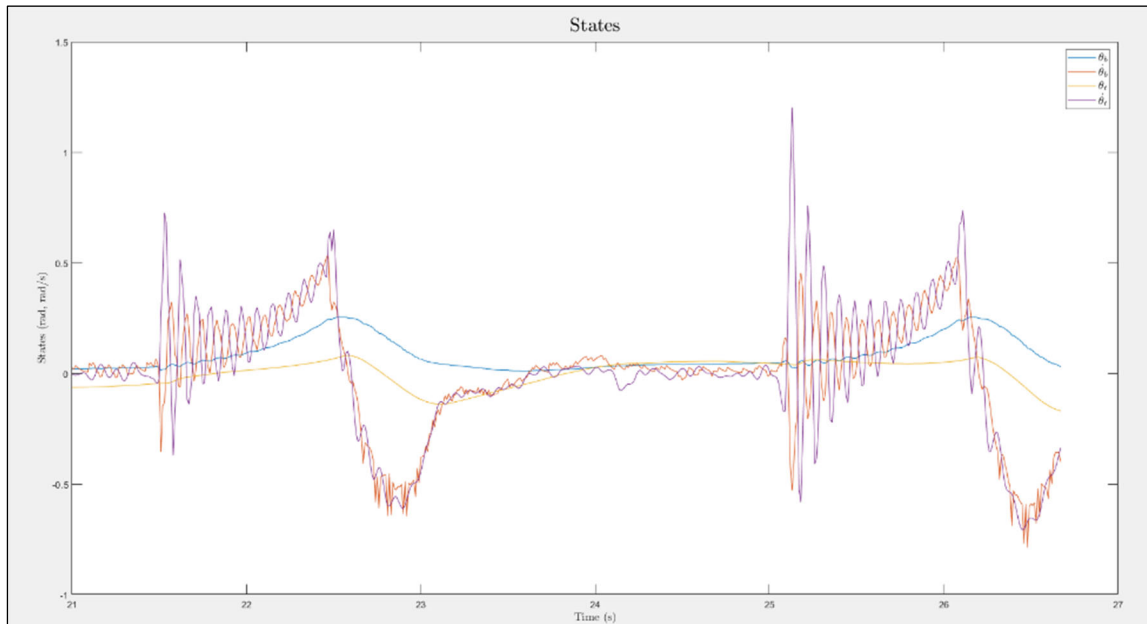


Figure 5.14 Penny FIP “pluck” test showing angular states for pendulum base and tip.

Here we note that, for a perfectly rigid inverted pendulum, the angles and angular rates between base and tip would agree. In the case of a flexible inverted pendulum, any real disagreement between base and tip angular states is reasonably assumed due to pendulum flex. This provides a powerful tool for flex disturbance mitigation, that is, unlike the superposition model developed in Section 5.3, the disparity between base and tip angular state data (5.33) provides a direct measurement of the systems flex state.

$$\delta_{\dot{\theta}} = \dot{\theta}_{base} - \dot{\theta}_{tip} \quad (5.33)$$

Note that the angular rate disparity between base and tip is not solely due to the pendulum's first flex mode, but a superposition of all flex modes. Figure 5.15 presents the flex state for the pluck test shown previously. The regular, periodic nature of the flex state during periods when it is freely oscillating (approx. 21.5 - 22.4 seconds, and 25.1 – 26.0 seconds) provides a convincing argument for the majority of the flex energy residing in the first oscillatory mode.

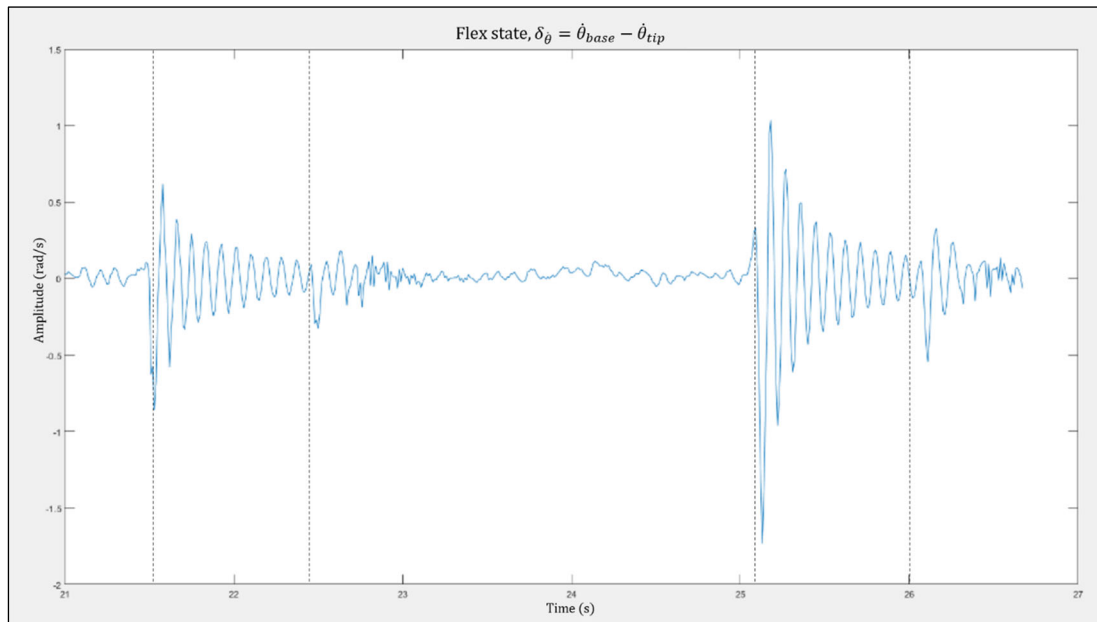


Figure 5.15 Penny FIP “pluck” test showing disturbance state.

The process described in Section 5.2 with parameter identification on measured data from tests with step inputs of amplitude 50 for 0.25 second resulted in a reasonable linear approximation of the system:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -42.9 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 7021 & -5131 & -3.28 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.4 \\ 0 \\ -71 \end{bmatrix}, \quad (5.34)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

with state vector  $[x \ \dot{x} \ \delta \ \dot{\delta}]^T$ . Figure 5.16 shows this model compared to a validation run using a doublet actuation command of amplitude 25 for 0.25 seconds and -25 for an additional 0.5 seconds.

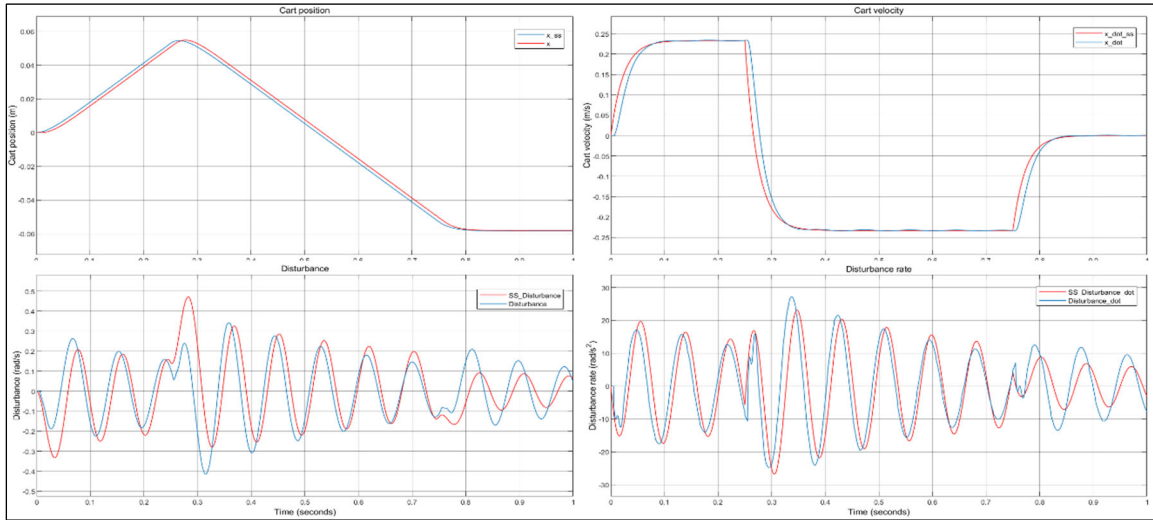


Figure 5.16 State-space model of FIP with disturbance state, compared to FIP.

As we proceed with descriptions of FIP testing and controller validation, the reader will note many FIP tests begin with a “doublet” excitation, meaning a step input followed immediately by another step input of opposite sign amplitude. The reason for this is that FIP is a very unstable dynamical system, and it is quite easy to drive the system past its

ability to recover. The doublet first pushes the pendulum to one side, then back toward equilibrium. In the open loop, this gives the system more time for measurements to be taken before toppling over. In the closed loop, the doublet excites the pendulum into oscillation without driving it unrecoverably far from equilibrium.

#### 5.4. Nonlinear Model Development

An initial first principles non-linear model was created in Simulink (see Figure 5.17) for a rigid inverted pendulum based on the equations described in Section 5.3.1 with the idea of expanding the model to include actuator and flexible pendulum dynamics. This approach was soon discarded in favor of Simscape modeling described in detail in the following section.

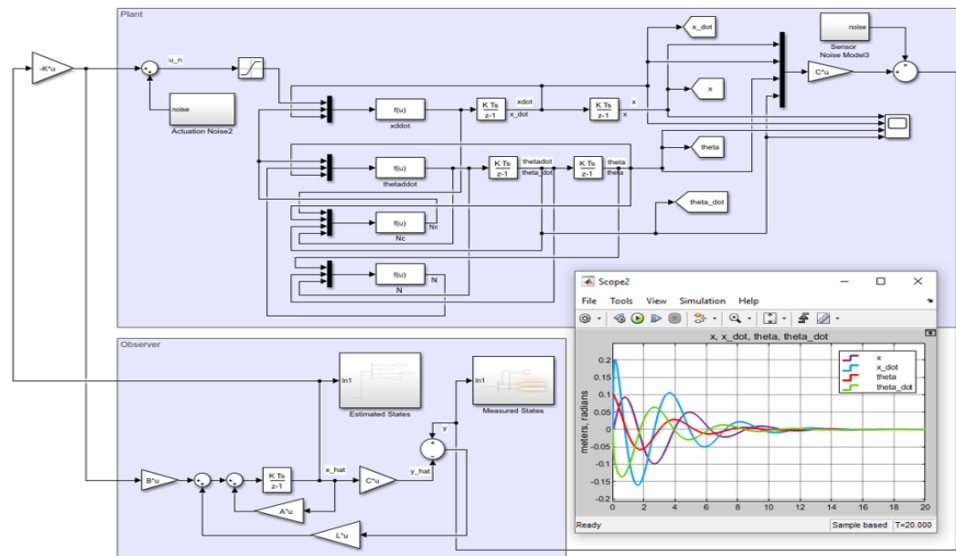


Figure 5.17 Nonlinear Simulink cart-pole model under linear control.

##### 5.4.1. Simscape model development

To implement actuator and flexible structure dynamics, Simscape was used for its pre-canned non-linear DC motor and structural models. Figure 5.18 shows the actuator dynamics model where the input voltage is applied to DC motor dynamics, producing

rotation of a friction parameterized tire. The force generated from the tire interaction with the ground is the output and can be applied to the cart model. DC gear motor and wheel parameters not available from the manufacturer were either empirically determined or determined such that simulated actuator behavior would match Penny operational data.

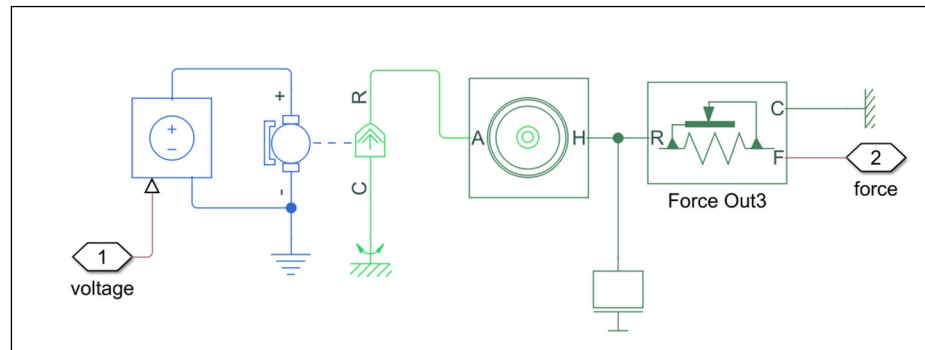


Figure 5.18 Simscape actuator dynamics model

With the input to the actuator modeled as voltage, functions were developed to reflect the nonlinearity of the hardware motor signal builder, allowing input command (cmd) signals in the same form as those that would be used on the real Penny system. This motor signal builder takes input cmd signals and outputs control (ctrl) signals in the range  $[-\gamma, \gamma]$ , where  $\gamma = 75$  is the value used to set hardware operational limits. The ctrl signal is then converted to voltage using the empirically determined relationship from Penny hardware testing,  $voltage = 0.1431 * ctrl$ . Modeling the motor signal builder in this way allows controllers developed in simulation to be easily transferrable to the hardware system.

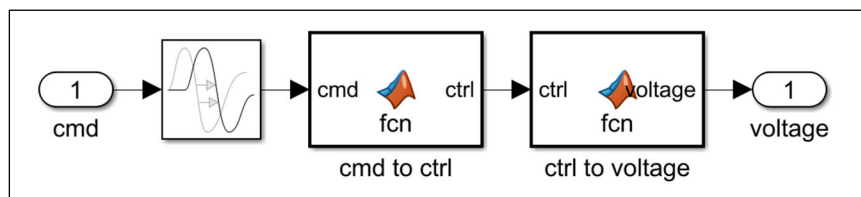


Figure 5.19 Simulink Motor Signal Builder mapping controller output to motor voltage.

Flexible structure models were implemented in Simscape using the built-in General Flexible Beam blocks. This beam model is fixed in the CFS case and attached to a Revolute Joint in the FIP case.

The additional dynamics imparted by the tip sensor hardware was also modeled, and the tip angle and angular rate states made available as outputs in the simulation using a Transform Sensor block. Once these were implemented in Simscape, it proved simpler and cleaner to replace cart dynamics with a Simscape representation as well. This resulted in the non-linear model for the CFS shown in Figure 5.20 and FIP in Figure 5.21.

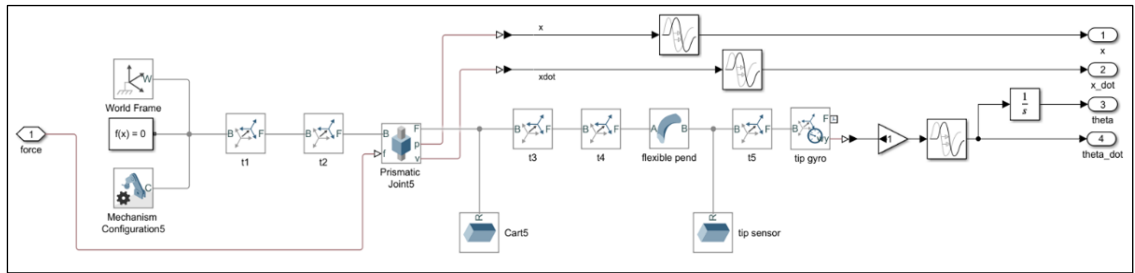


Figure 5.20 – Simscape nonlinear dynamics model of CFS

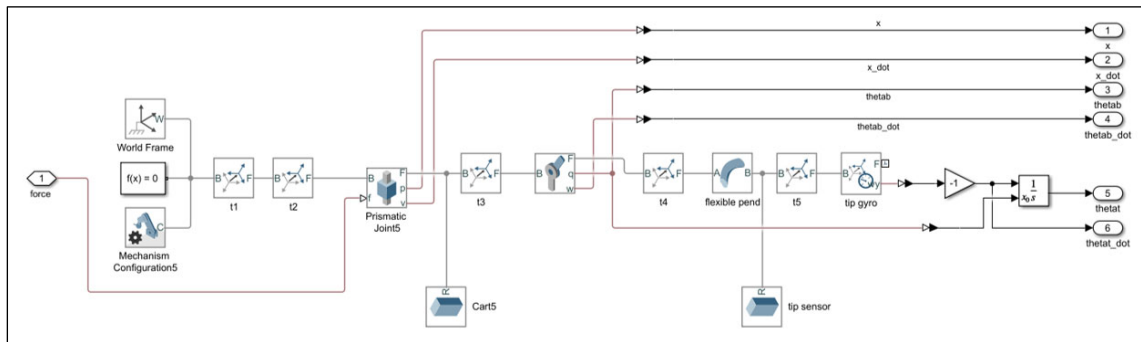


Figure 5.21 – Simscape nonlinear dynamics model of FIP

These models produced cart dynamics in close agreement with recorded Penny data. However, oscillatory dynamics were apparently different in both CFS and FIP cases as simulated frequencies were higher than observed on Penny and attenuation noticeably

slower. It was believed that cumulative errors in published and measured parameters were the root cause.

Since CFS and FIP oscillatory parameters were closely quantified (reference Section 5.1), the simulation's oscillatory dynamics were subsequently tuned such that the natural frequency and damping ratios of the flexible structures closely matched Penny by adjusting pendulum parameters within their General Flexible Beam blocks.

Figure 5.22 compares the output of the CFS non-linear model to the output of the CFS linear model developed previously. The close agreement between linear and non-linear models is convincing (recall the close agreement between linear model and hardware shown previously).

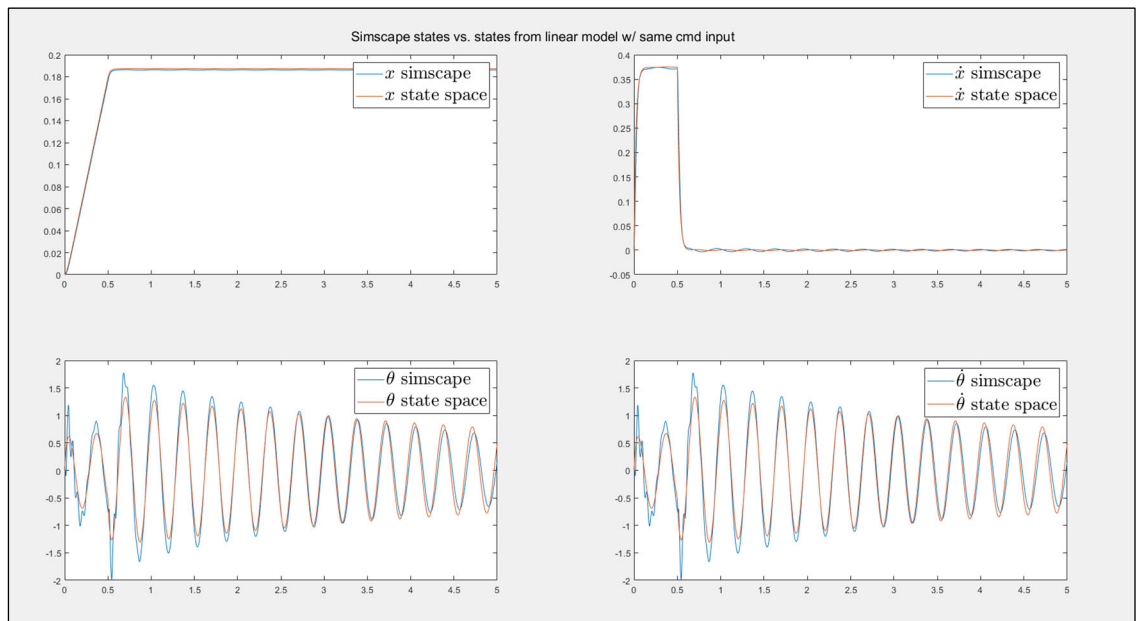


Figure 5.22 Resulting linear model vs. Simscape model for CFS.

Figure 5.23 shows a comparison of angular rate data between Penny FIP's pendulum pluck test data and its Simscape model after parameter tuning was complete.

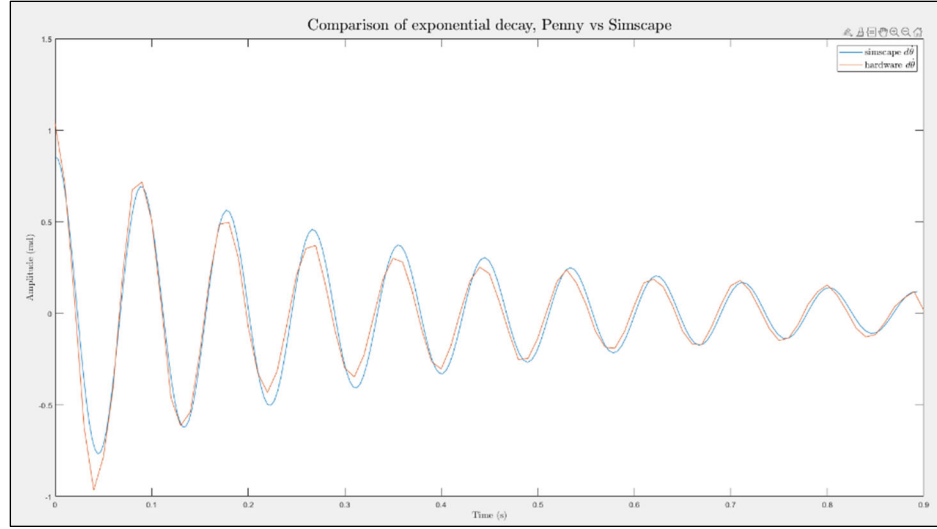


Figure 5.23 Comparison of  $\dot{\theta}_b$  state, simulation vs. FIP.

### 5.5. FIP Linear Model Development from Non-Linear Simulation

A final linear model was developed based on the dynamics of the nonlinear Simscape simulations, which had been parameterized to closely match Penny FIP outputs, using the same process as with the hardware. This model was produced because there are differences between hardware and the nonlinear simulations, so new linear models based solely on the linearized dynamics of the Simscape output were produced to be as representative as possible of the nonlinear models. This new state space representation models only FIP's rigid body dynamics for the purposes of linear analysis and linear controller design. The rationale for this approach is that subsequent disturbance accommodation is performed by treating the system's oscillatory dynamics as the system's disturbance, subsequently accommodating it in feedback. This is done by augmenting a system already under linear control whose linear controller was tuned for rigid body dynamics (not designed for flex disturbance accommodation).

To produce this model, the same process was followed as in Section 5.2, where data sets were recorded by applying command step inputs ( $u$ ) to the nonlinear model this time

and recording the relevant state data. The MATLAB system identification toolbox was again used to generate independent transfer functions for each input-output relationship:

$$\begin{aligned} u &\rightarrow x \\ u &\rightarrow \dot{x} \\ u &\rightarrow \theta \\ u &\rightarrow \dot{\theta} \end{aligned} \quad (5.35)$$

These transfer functions were then used to record linearized state data given the same simulated step input (reference Figure 5.3). The MATLAB system identification toolbox was again used to generate a state space model from the linearized recorded data. The process described above resulted in the following linear model for the Simscape FIP system:

$$\begin{aligned} A_r &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -42 & -0.9 & 0.9 \\ 0 & 0 & 0 & 1 \\ 0 & 29.7 & 7 & 0 \end{bmatrix}, B_r = \begin{bmatrix} 0 \\ 0.4 \\ 0 \\ -0.28 \end{bmatrix}, \\ C_r &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, D_r = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (5.36)$$

Figure 5.24 shows the output of the Simscape and state space (linear) models given the same actuation command of 75 for 0.10 seconds. This step actuation command and duration was shorter than for CFS because the pendulum angle in FIP configuration will rapidly diverge in the absence of stabilizing control.

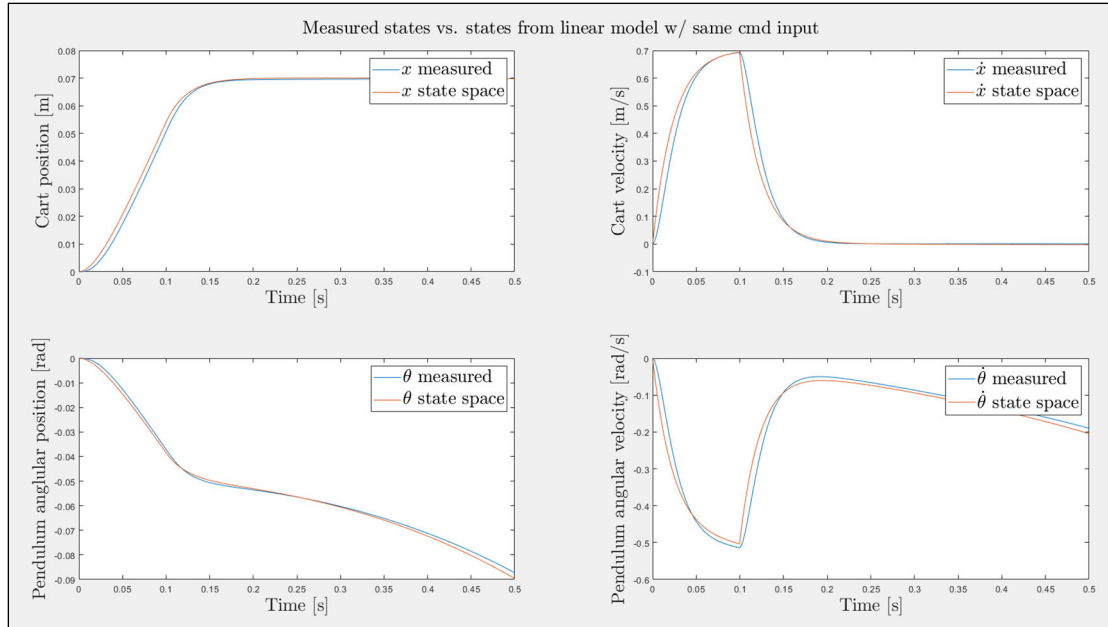


Figure 5.24 Resulting Linear Rigid Body Model vs. Simscape Model for FIP

Figure 5.25 and Figure 5.26 show the linear model versus the Penny FIP rigid body state responses for the same step input of 75 for 0.10 seconds. Other than the hardware time delay, the close agreement between this new linear model and the Penny FIP rigid body responses provides a fair argument for using the linear model in (5.36) henceforth for the application of linear systems analysis and control theory.

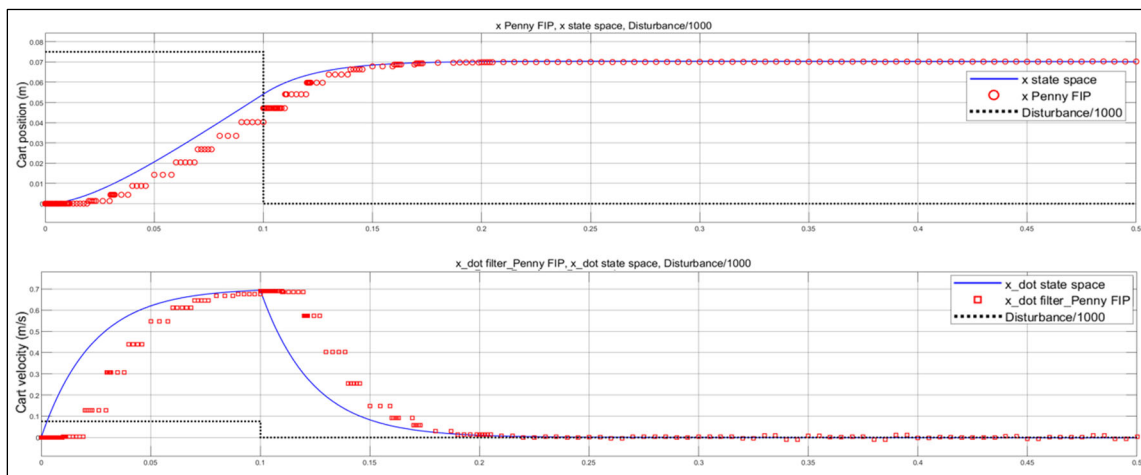


Figure 5.25  $x$ -states comparison between Linear Rigid Body Model and Penny FIP.

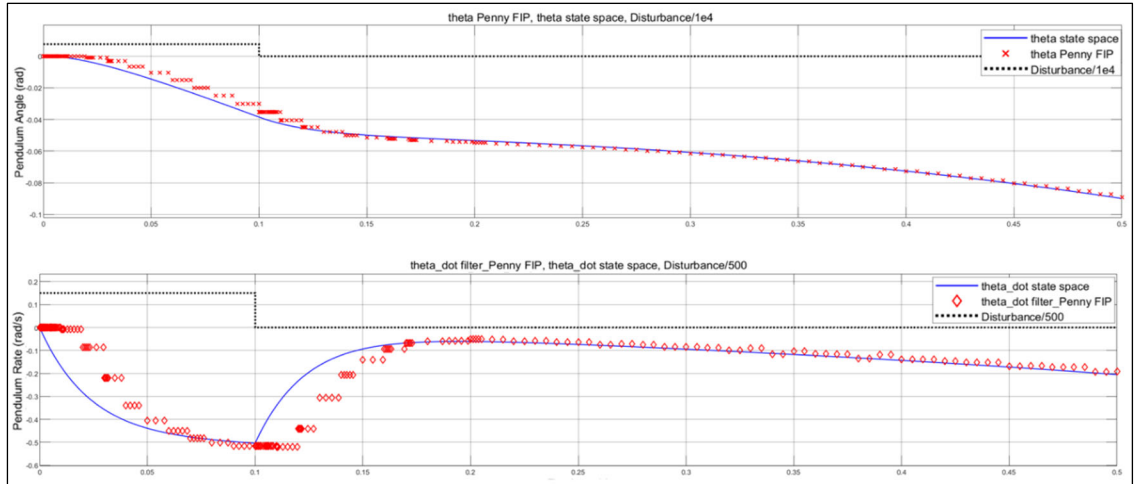


Figure 5.26 Theta-states comparison between Linear Rigid Body Model and Penny FIP.

## 6. Controller Development and Application on Hardware

CFS and FIP linear time-invariant models are now used to design linear controllers for state regulation and control of Penny’s rigid body states (the “nominal plant”). These are the cart states  $(x, \dot{x})$  for Penny CFS, and the rigid body states  $(x, \dot{x}, \theta, \dot{\theta})$  for Penny FIP. Controller designs were first tested on the high-fidelity nonlinear simulation before deployment to the actual hardware. Each test is summarized here, although it was the result of many simulations and hardware testing cycles.

### 6.1. CFS Rigid Body Control

Recall from Section 5 that the Penny CFS system is described by (5.9) with full state knowledge. For the real Penny CFS system, we only have measurements for cart position  $(x)$  and beam tip angular velocity  $(\dot{\theta})$ . The LTI state-space model for this system therefore becomes

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -22.5 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 33.1 & -74.1 & -0.092 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0.2 \\ 0 \\ -0.34 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (6.1)$$

with state vector  $[x \ \dot{x} \ \theta \ \dot{\theta}]^T$  and Laplace-domain open-loop transfer functions:

$$TF_{u \rightarrow x} = \frac{0.2 s^2 + 0.0184 s + 14.82}{s^4 + 22.59 s^3 + 76.17 s^2 + 1667 s} \quad (6.2)$$

$$TF_{u \rightarrow \theta} = \frac{-0.34 s^3 - 1.03 s^2}{s^4 + 22.59 s^3 + 76.17 s^2 + 1667 s} \quad (6.3)$$

We first bring Penny’s cart under linear regulation using a Separation Principle controller/observer described in Section 2.1. From the state evolution matrix in (6.1) we

note this linear model's nominal plant states  $(x, \dot{x})$  possess no  $\theta$  terms, so we can write the cart system's dynamics as:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ 0 & -22.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.20 \end{bmatrix}, \\ C &= [1 \quad 0], \quad D = [0], \end{aligned} \tag{6.4}$$

with state vector  $[x \ \dot{x}]^T$  and Laplace-domain open-loop transfer function:

$$TF_{u \rightarrow x} = \frac{0.2}{s^2 + 22.5s} \tag{6.5}$$

The closed-loop poles are now arbitrarily placed. After some brief trial and error, placing the cart's poles at  $p = [-4 + i, -4 - i]$  results in good system performance. This was done using Matlab's "place" function on  $(A, B, p)$ , resulting in the controller gain matrix:  $K = [85 \quad -72.5]$ .

To obtain the cart velocity state, we could just differentiate and filter the position signal, which was often done when collecting cart state data for modeling purposes. Here we use a full state observer to estimate any state not measured directly. Since we need the observer's estimates to converge much faster than the controller, we purposefully place observer poles far to the left of the  $j\omega$ -axis. After some trial and error, we find that good performance is elicited by placing the observer's poles at  $l = [-200 + 50i \quad -200 - 50i]$  again using Matlab's *place* function on matrices  $[A^T, C^T, p]$ , resulting in the observer gain matrix:  $[377 \quad 3.4e4]^T$ . The system design is shown in Figure 6.1.

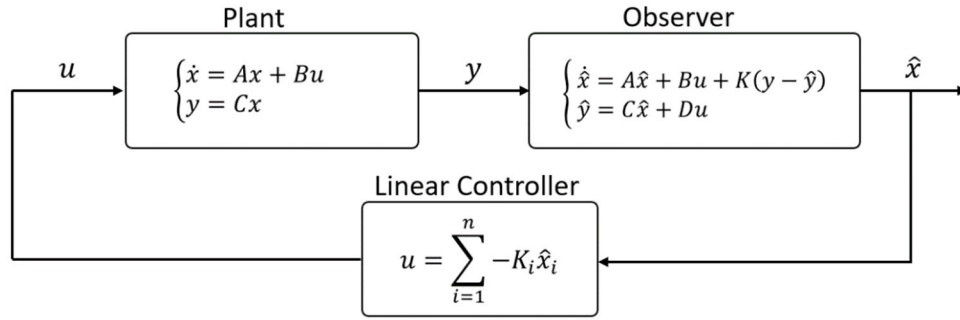


Figure 6.1 Penny CFS linear control with separation principle observer-controller.

Figure 6.2 shows the control and monitoring setup in Simulink, with observer expanded here for clarity. Future figures will collapse complex systems into their own “subsystems” for space efficiency as was done for the plant model (upper left) and scopes (upper right). The Plant model is the motor command and data acquisition setup described in Section 4.2, only without pendulum base angle measurements.

Note the step function generator at the far left in Figure 6.2. This is used to excite the system to test observer/controller performance. Note also that its output is available to the observer. This is the case for setup and tuning purposes, so the observer is not working with state responses that were not produced by the control signal it receives. This excitation block will later be moved downstream of the observer, so the observer has no knowledge of external excitations not created by the control law.

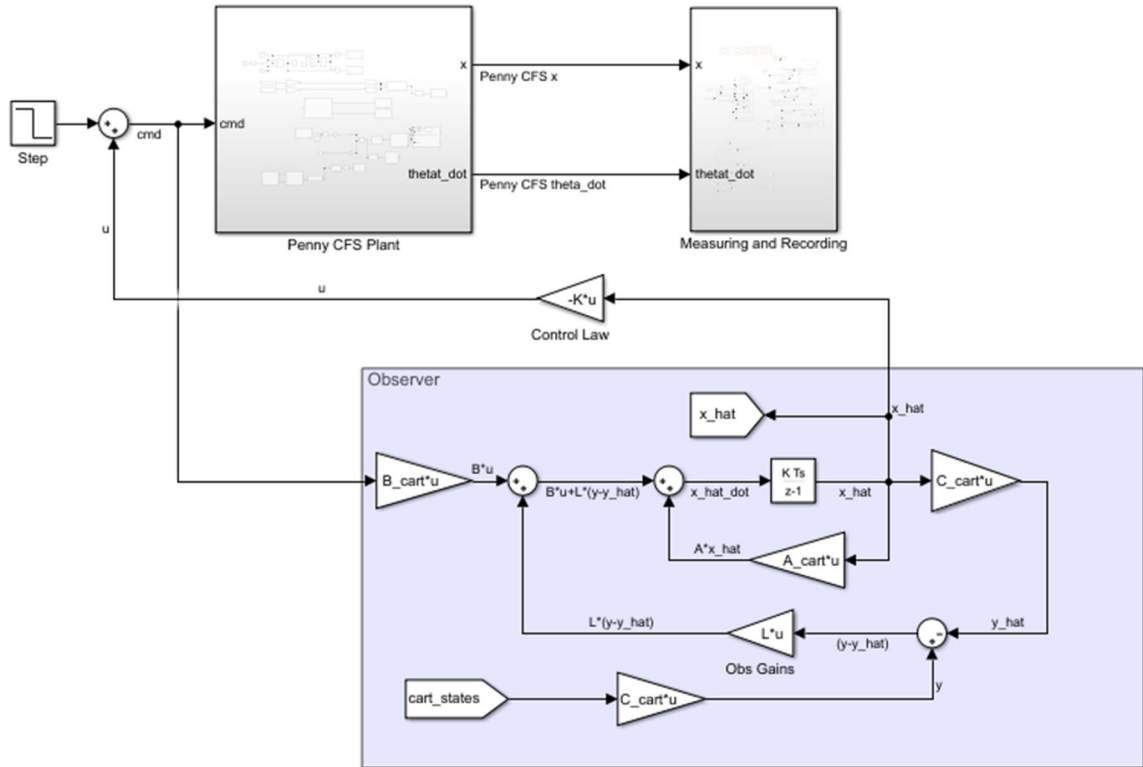


Figure 6.2 Simulink CFS model with separation principle observer-controller.

Figure 6.3 shows the cart's measured and estimated states in response to a step command of magnitude 25 for 0.5 seconds. Observer states were initialized to  $[0 \ 0]$ . When a state label includes the moniker “\_hat,” it indicates the estimate of that state. Also, note the control law is operating on the estimated cart position  $\hat{x}$  when it could be fed the direct measurement  $x$ . It would be fed  $x$  if state convergence was slow and tracking poor, but this is currently not the case.

Although the observer is not currently estimating the beam's angular states, the system's  $\dot{\theta}_t$  measurement is still shown in Figure 6.4 for the reader's information.

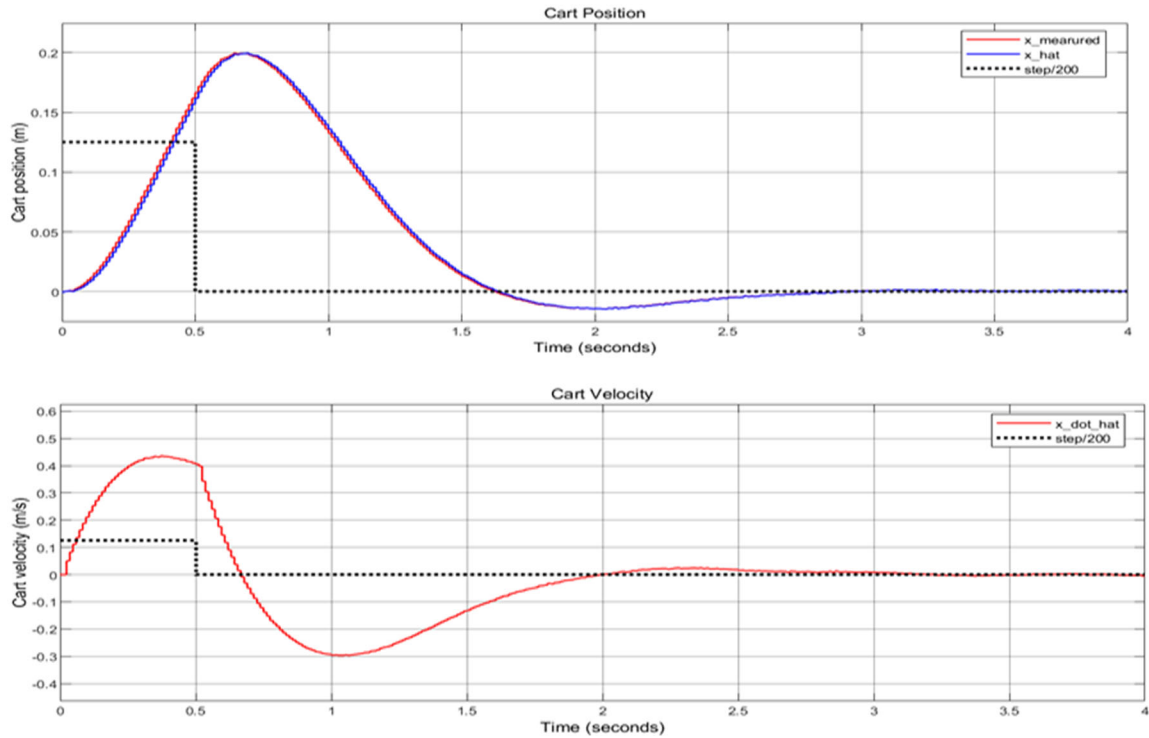


Figure 6.3 Controller/observer state responses to step input.

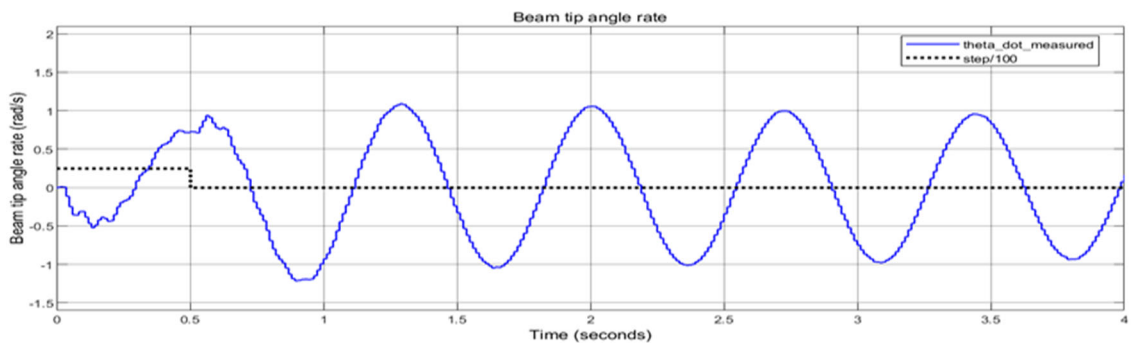


Figure 6.4 Beam flex state responses to step input.

Now that the cart's observer/controller has been tuned, it is presented with a plant disturbance it does not “see” in the control signal (reference Figure 6.5). The step disturbance is still of magnitude 25 for 0.5 seconds as before.

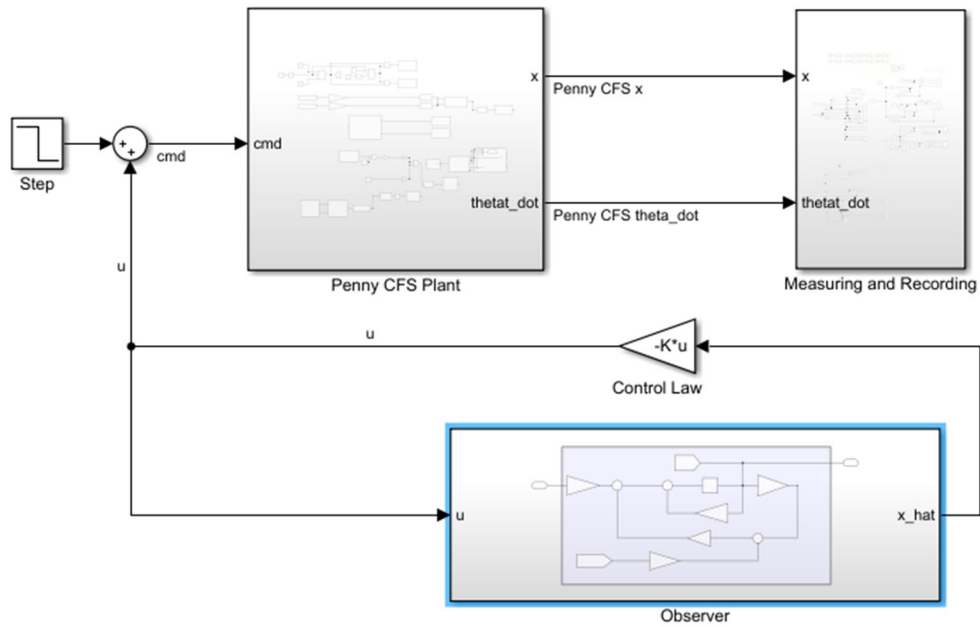


Figure 6.5 CFS control model with “unseen” disturbance.

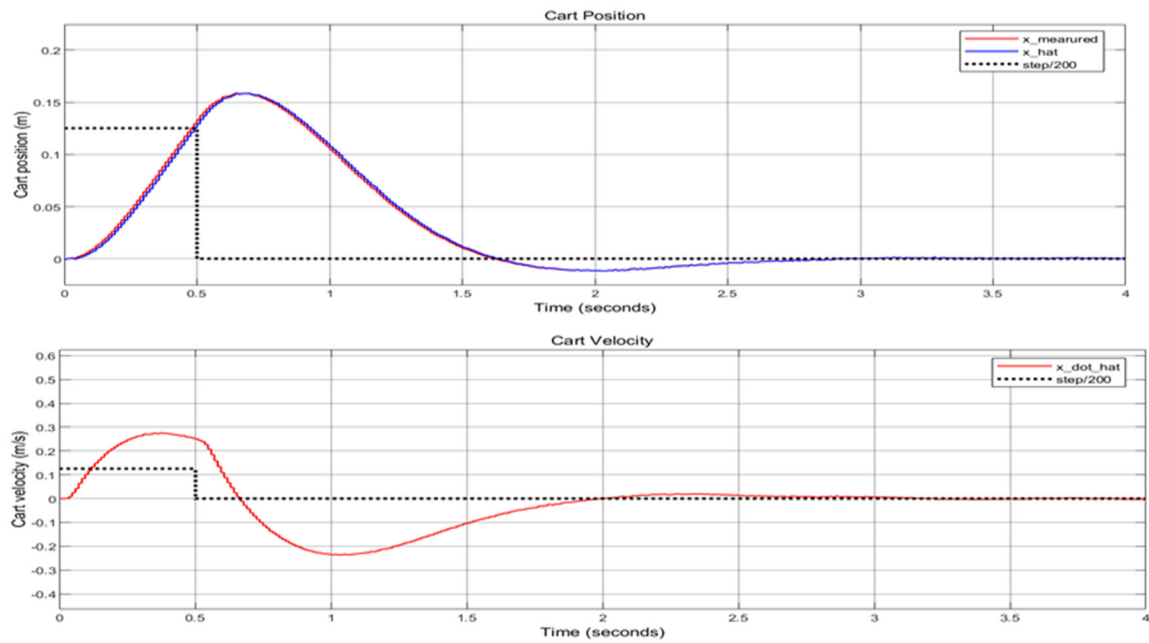


Figure 6.6 Cart state response to “unseen” disturbance.

## 6.2. Penny CFS Augmentation by Adaptive Regulation

Now the system's internal oscillatory dynamics are treated as the “plant disturbance” and several mitigation approaches are taken. First, the system is augmented with adaptive output regulation of the form shown in Figure 6.7.

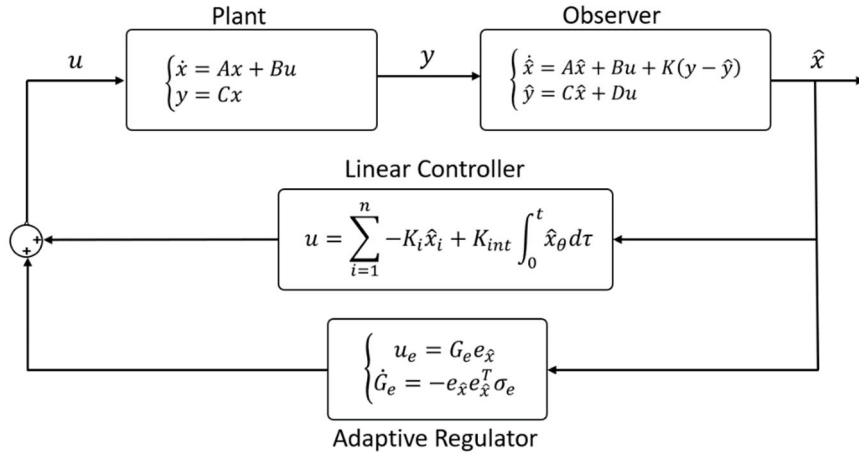


Figure 6.7 Schematic of augmentation for adaptive regulation.

The adaptive augmentation is now added to the Simulink control model to implement the adaptive control law as shown in Figure 6.8.

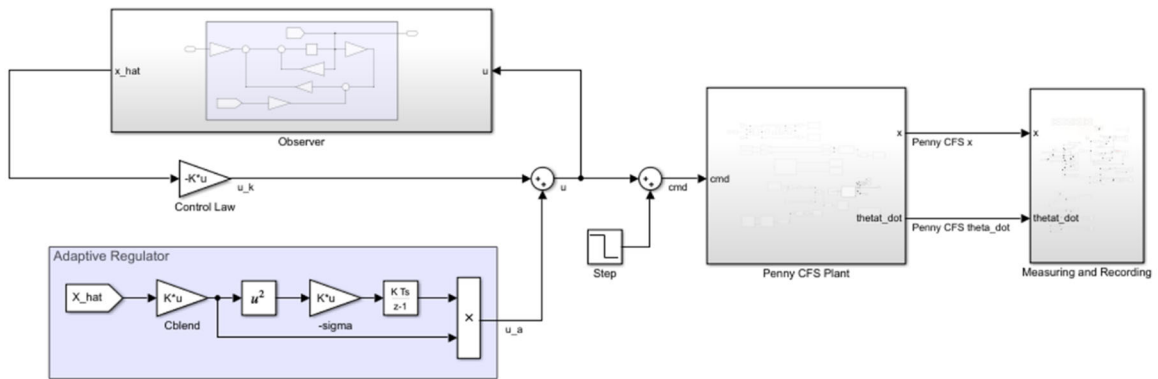


Figure 6.8 – Simulink CFS control model augmented with adaptive regulation.

Note that the numerator of the system's transfer function  $u \rightarrow \theta$  from (6.3) is not entirely stable, with zeros residing at  $[0, 0, -3.03]$ , meaning the system does not meet

the ASPR requirement for direct adaptive augmentation. To guarantee asymptotic state convergence, the zeros were placed through sensor blending (reference Section 2.5) such that the plant appears almost strictly positive real. Many combinations of zeros and adaptive gains ( $\sigma_e$ ) were implemented and a wide variety are effective in mitigating the beam oscillation. Figure 6.9 shows the measured state responses for unaugmented (red) and augmented (blue) systems when the open loop transfer function zeros are placed at  $Z = [-5, -6, -7]$  with an adaptive gain  $\sigma_e = 1.4$ , creating the sensor blending matrix  $C_{blended} = [14.17 \ 4.22 \ -42.98 \ -0.462]$ .

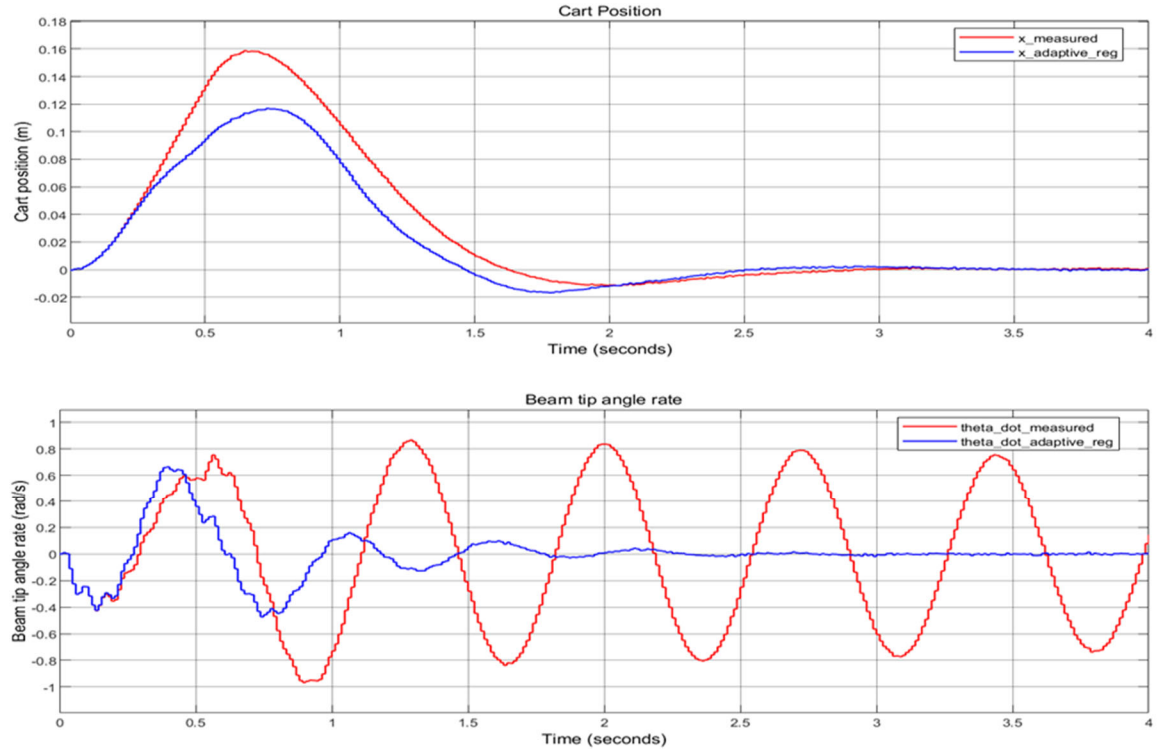


Figure 6.9 Penny CFS under unaugmented (red) and augmented (blue) control.

To gauge the relative “cost” to the system of this augmentation, the cumulative effort was computed for each test (see Figure 6.10), defined as the integral of the absolute value of the command signal. Ideally, power consumption would be used as a quantitative

metric, but Penny does not currently have the sensors needed to record this data, only LED displays for real-time voltage and current draw.

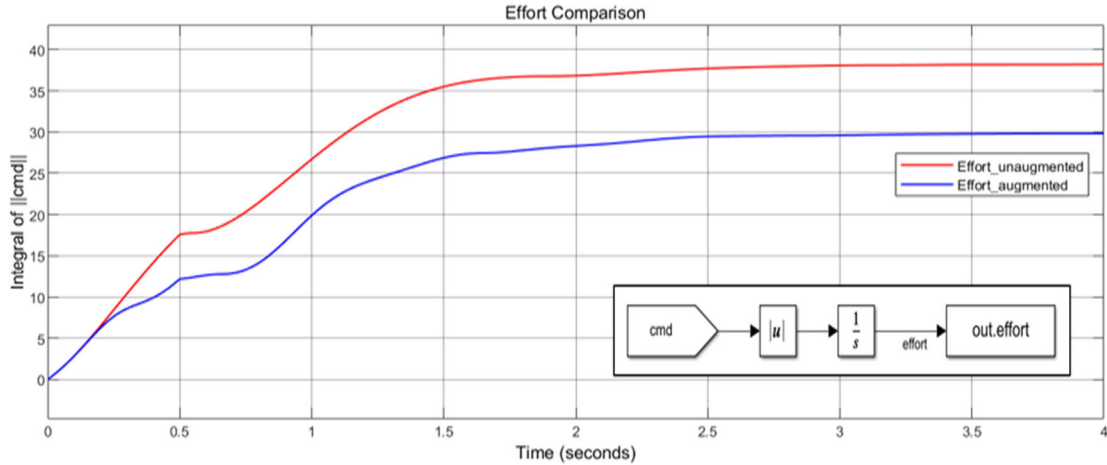


Figure 6.10 Cumulative effort, unaugmented and augmented control (equation inset).

Note that no claims are being made as to the superiority of the adaptive augmentation approach over other methods. Other controllers applied to the disturbance state would doubtless effectively manage system flex dynamics and control architecture comparisons are an area of future work. This investigation seeks to validate the current approach on flexible structure hardware as one potential means of augmentative flex dynamics mitigation.

### 6.3. Penny CFS Augmentation by Disturbance Accommodation

The Penny CFS system, previously augmented with adaptive output regulation, is now given the additional disturbance accommodation augmentation shown in Figure 6.11.

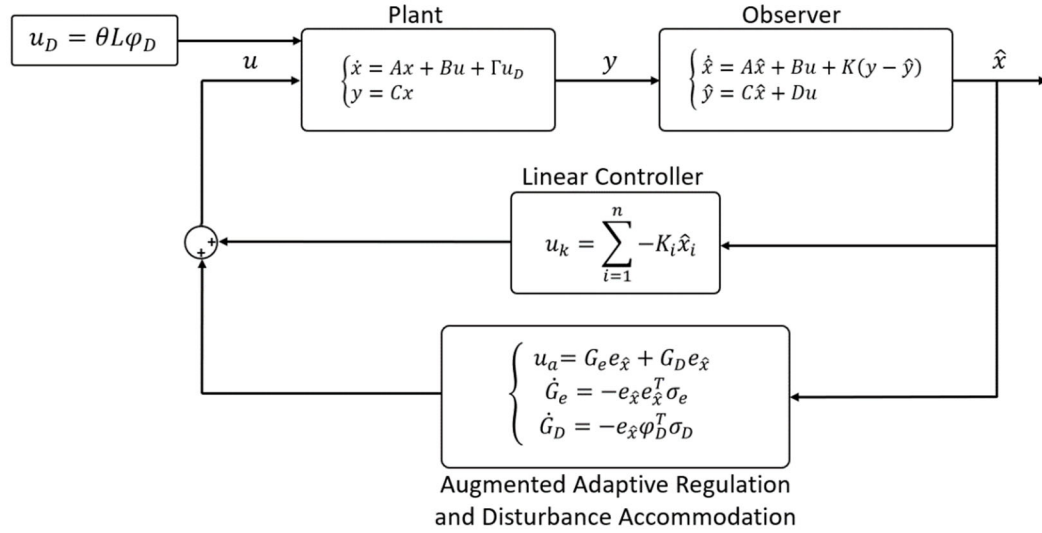


Figure 6.11 Augmentation for adaptive regulation and disturbance accommodation.

The Simulink UI control model is again updated to implement the adaptive control law as shown in Figure 6.12. Recall from (2.7) that the operator  $\varphi_D$  contains the basis functions of the disturbance of known form but unknown magnitude. In the case of a sinusoidal disturbance of known frequency,  $\omega_n$  of the beam in this case,  $\varphi_D = [\sin(\omega_n t), \cos(\omega_n t)]$ .

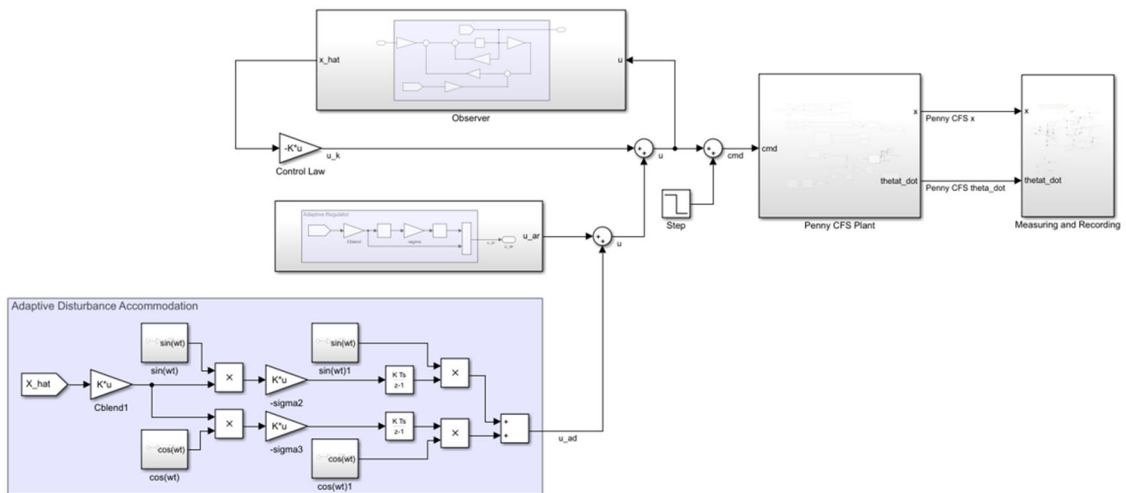


Figure 6.12 Penny CFS with adaptive regulation and disturbance accommodation.

Tuning of the adaptive disturbance gain found that  $\sigma_D = 0.22$  resulted in good performance without introducing instability. As the cantilevered structure's first vibrational mode is mitigated, note the emergence of the second mode in the measured states from 0.5 – 1.5 seconds shown in Figure 6.13. The controller effort comparison is again shown in Figure 6.14.

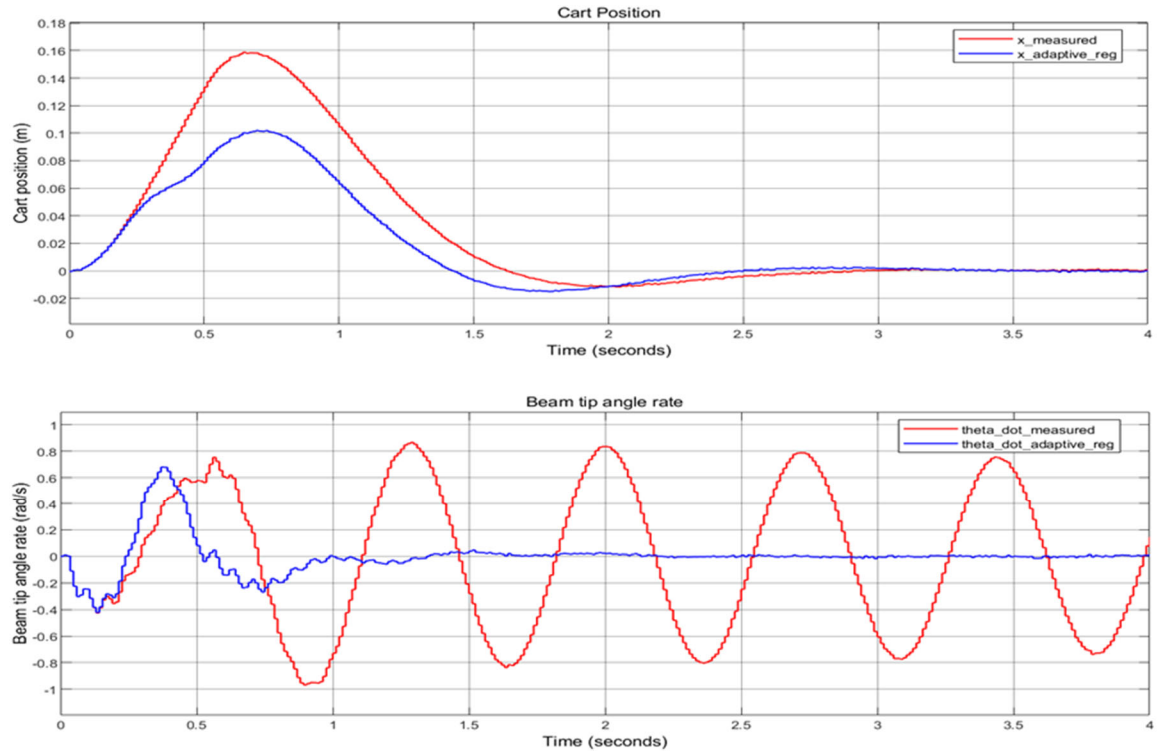


Figure 6.13 Penny CFS measured states under unaugmented (red) and augmented (blue) adaptive regulation and sinusoidal disturbance accommodation.

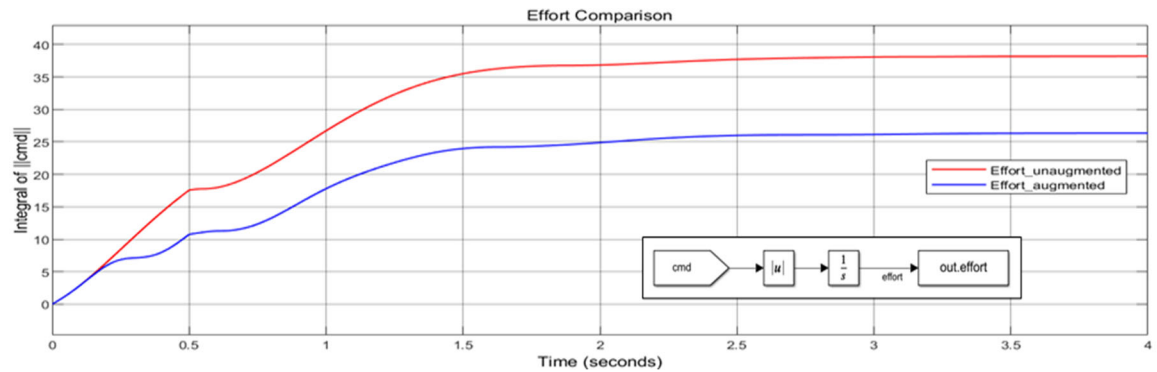


Figure 6.14 Cumulative effort, unaugmented and augmented control (equation inset).

#### 6.4. FIP Rigid Body Control

Recall from Section 5 that the Penny FIP system is described by (5.36) when full state knowledge is present. For the real hardware system, we only have measurements for cart position ( $x$ ) and pendulum base angle position ( $\theta$ ). The LTI state-space model for the physical system, with revised output matrix, is therefore

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -42 & -0.9 & -0.9 \\ 0 & 0 & 0 & 1 \\ 0 & 29.7 & 7 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0.4 \\ 0 \\ -0.28 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (6.6)$$

with state vector  $[x \ \dot{x} \ \theta_b \ \dot{\theta}_b]^T$  and Laplace-domain open-loop transfer functions:

$$TF_{u \rightarrow x} = \frac{0.4s^2 - 0.25s - 2.548}{s^4 + 42s^3 - 33.73s^2 - 267.27s} \quad (6.7)$$

$$TF_{u \rightarrow \theta} = \frac{-0.28s + 0.12}{s^3 + 42s^2 - 33.73s - 267.27} \quad (6.8)$$

We now design and implement a linear controller for Penny FIP's rigid body dynamics, since the rigid brace described in Section 3 is used to eliminate slow (measurable) flex dynamics. A Separation Principle Controller was first attempted, but no amount of tuning resulted in adequate stabilization. Unmodeled dynamics are believed to be the cause. Ultimately an integrator was added to help manage rigid body dynamics, resulting in satisfactory performance, i.e., balancing the rigid pendulum indefinitely. As before, a linear controller is designed to manage the nominal plant, the rigid body system in this case, so the disturbance term is not addressed by linear control.

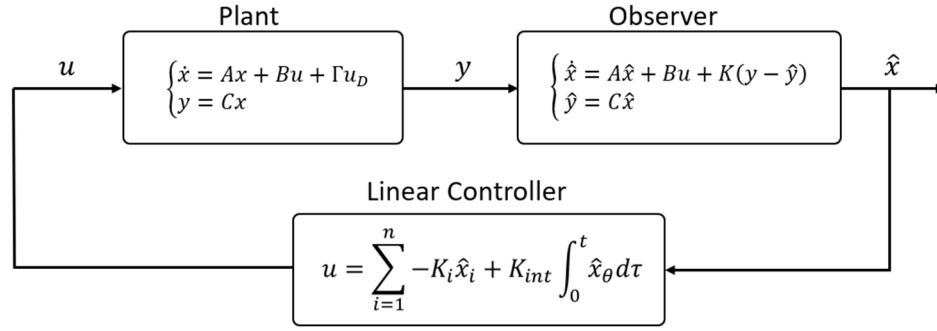


Figure 6.15 Diagram of Penny FIP system under linear control.

The closed-loop poles were again placed using Matlab's "place" function on  $(A, B, p)$  throughout a long tuning campaign. In combination with the integral control, good rigid body performance was achieved placing the closed-loop poles at  $p = [-1, -1.1, -55, -9]$  resulting in the controller gain matrix:  $Kc = [-213.7 \ -638.5 \ -2531 \ -998.3]$ . Observer poles were placed at  $l = [-115, -121, -6325, -1035]$  resulting in the following observer gains:

$$Lo = \begin{bmatrix} 6144 & 436320 & 1331 & 375430 \\ 999.1 & 798440 & 1416 & 195830 \end{bmatrix}^T \quad (6.9)$$

In Figure 6.16 the integrator is implemented using a PID control block. Note that only the integrator parameter is being utilized in this block, with a gain  $K_{int} = -2250$ .

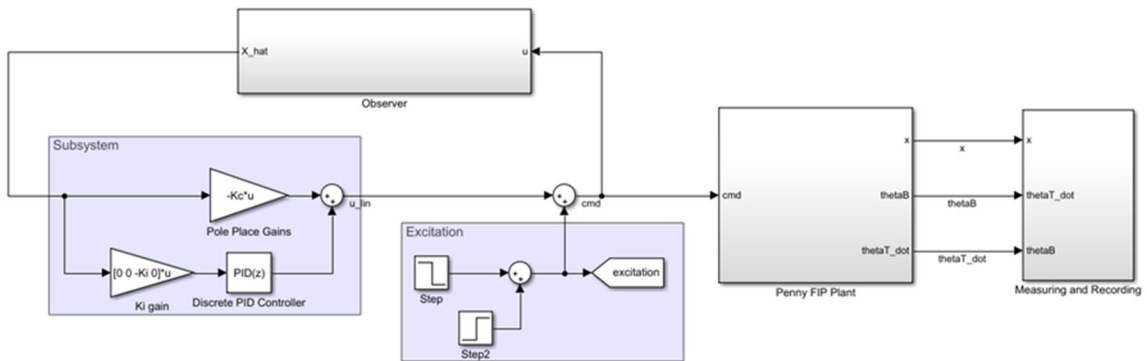


Figure 6.16 Penny FIP control model with linear controller and excitation.

The following figures highlight linear controller performance when a doublet of amplitude  $\pm 10$  is injected. Again, this type of excitation is helpful for inverted pendula which cannot diverge too far from equilibrium before becoming unrecoverable.

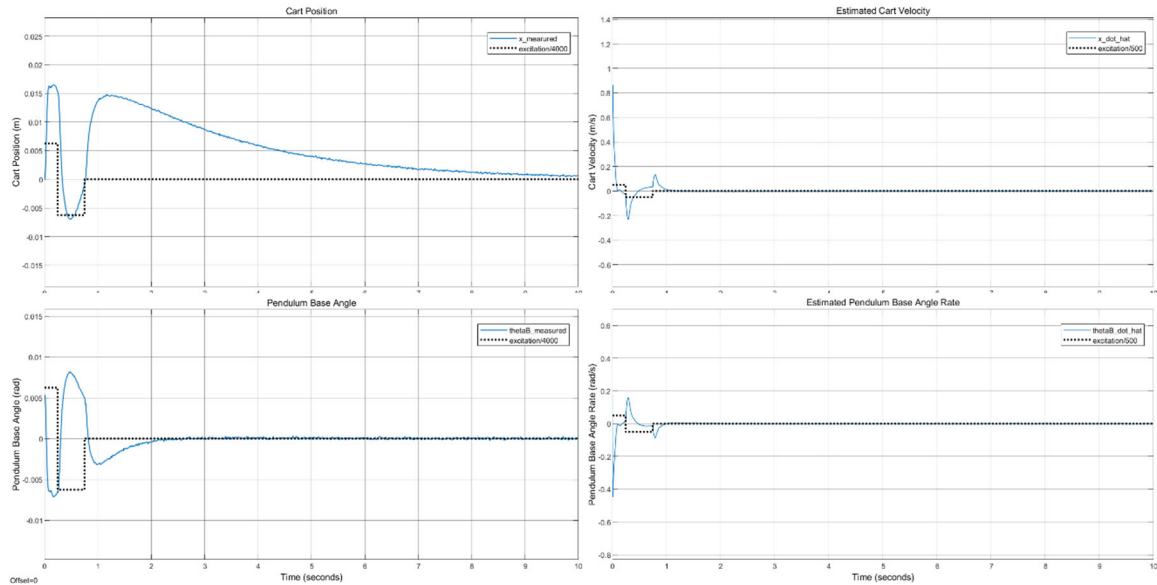


Figure 6.17 Penny FIP rigid body measured states under linear control (10 sec).

The position state is slow to converge, but as its peak departure was less than two centimeters, the gains are left alone. We now expand the first few seconds for clarity.

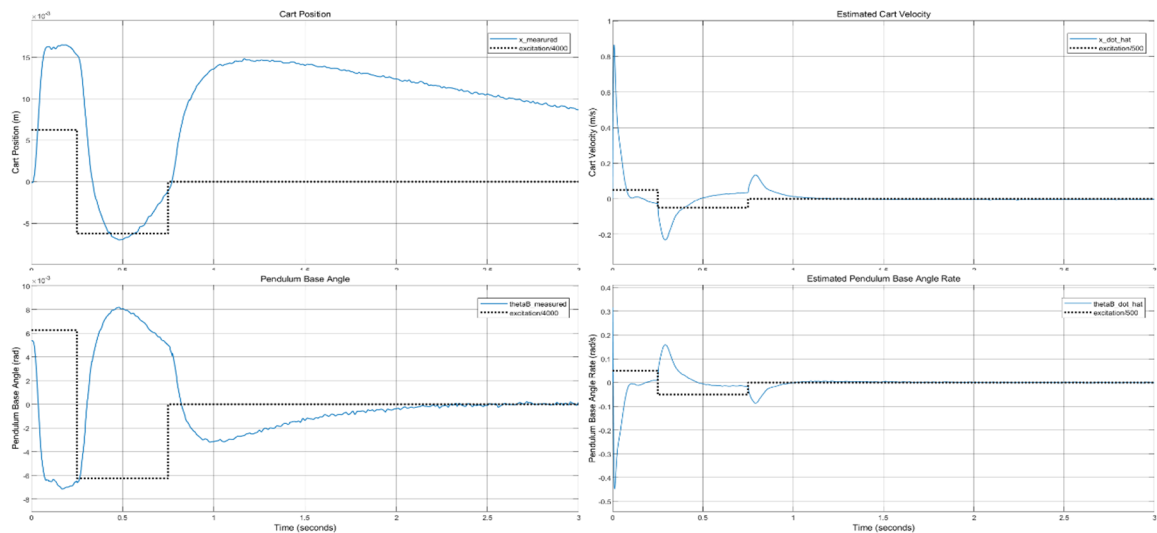


Figure 6.18 – Penny FIP rigid body measured states under linear control (3 sec).

### 6.5. FIP Augmentation with Direct Adaptive Regulation

From the previous test onward, the estimates of the measured states will no longer be used since Penny FIP's instability and fast flex dynamics have proven more difficult to manage using state estimates when measurements are readily available for  $x$  and  $\theta$ . Figure 6.19 presents this new architecture. Again, future models will collapse the highlighted systems into subsystem blocks for space efficiency.

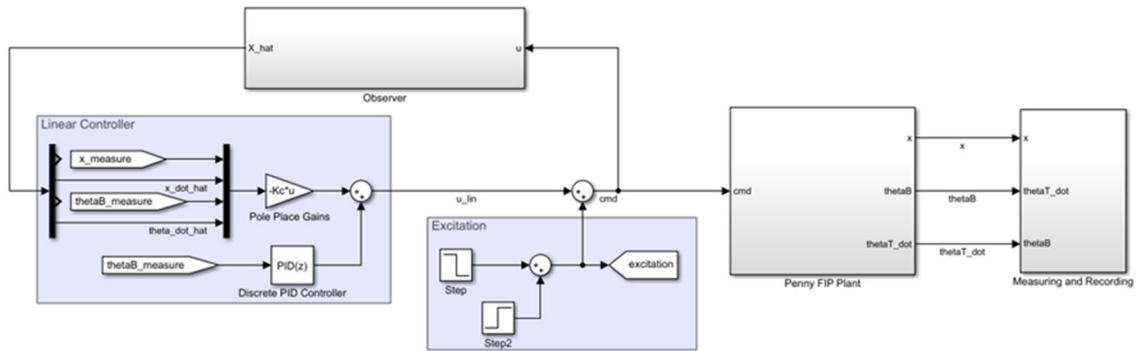


Figure 6.19 Penny FIP linear control model with flex present.

Before removing the pendulum's braces and allowing the pendulum to flex, we first implement adaptive augmentation on the rigid body (reference Figure 6.20).

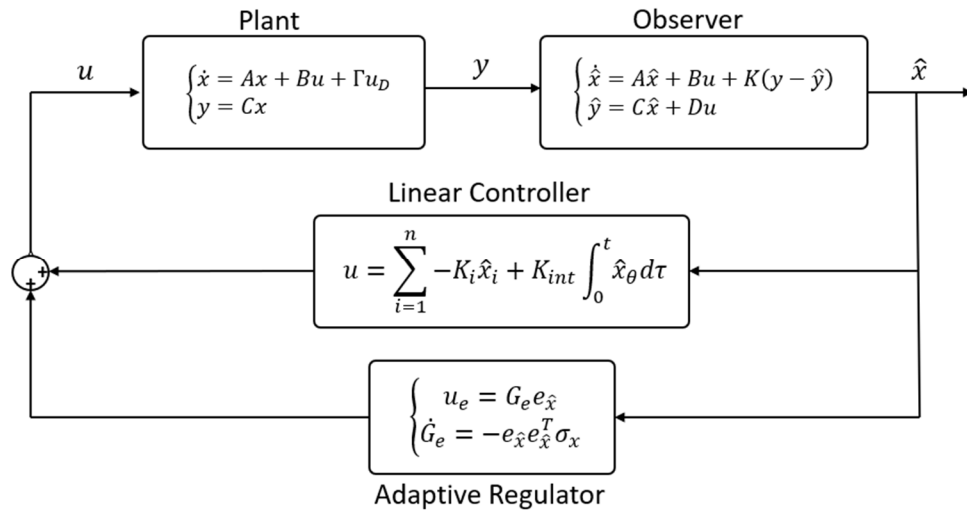


Figure 6.20 Penny FIP with rigid plant augmented with adaptive regulation.

We see from (6.8) that the system's open-loop transfer function is not ASPR since its zeros reside at  $[0, 0.429]$ , nor is its high-frequency gain strictly positive real. Sensor blending was again employed to make the open-loop plant appear ASPR to the adaptive regulator. As before, many combinations of zeros and adaptive gains ( $\sigma_e$ ) were implemented and a wide variety are effective in output regulation. Figure 6.24 shows the measured state responses for a representative (see Results discussion) test of the unaugmented (red) and augmented (blue) systems when the open loop transfer function zeros are placed at  $Z = [-0.1, -1, -30]$ , with an adaptive gain  $\sigma_e = 2200$ , creating the sensor blending matrix  $C_{blended} = [-1.2 \ -18.0 \ -109.1 \ -29.3]$ . The adaptive regulator used is structurally identical to the one shown in Figure 6.8.

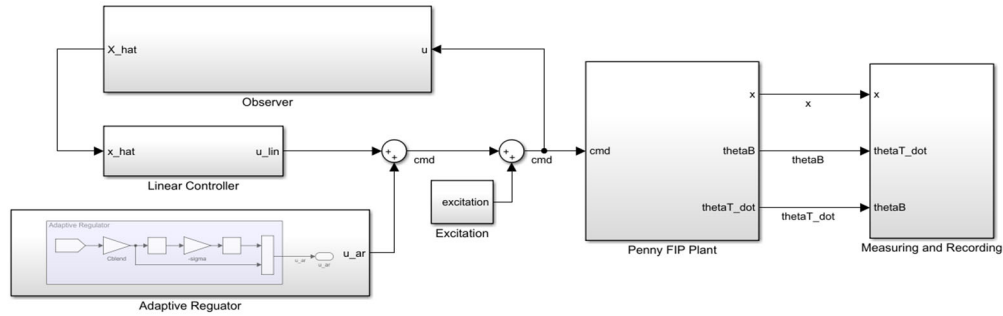


Figure 6.21 Penny FIP control model augmented with adaptive regulation.

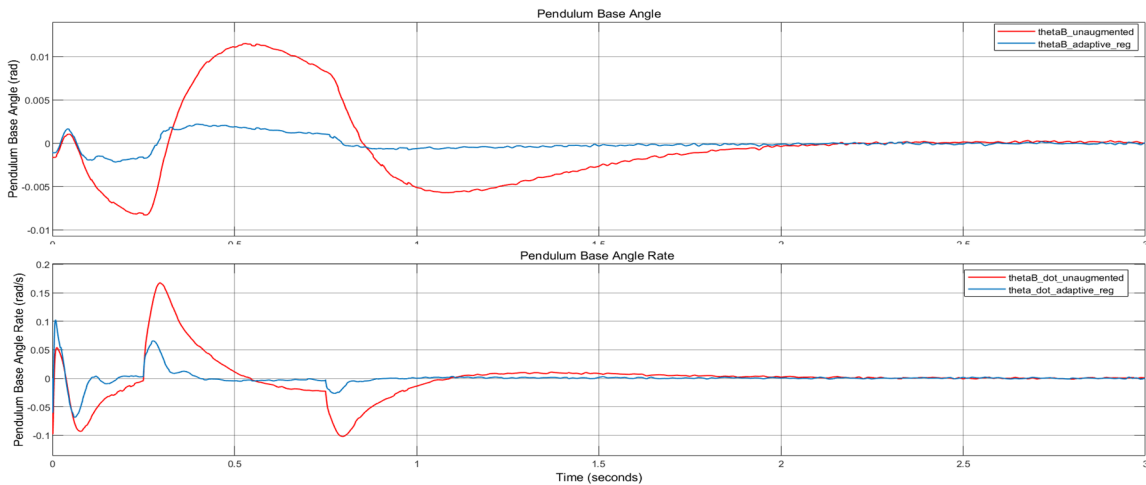


Figure 6.22 Penny FIP angular states for rigid plant augmented with adaptive regulation

The rigid braces are now removed from Penny FIP’s long Aluminum pendulum and the performance of the linear controller is demonstrated in the presence of flex dynamics. For this test, the adaptive regulator is disabled. Figure 6.23 shows ten seconds of state response to the doublet excitation of amplitude  $\pm 25$ .

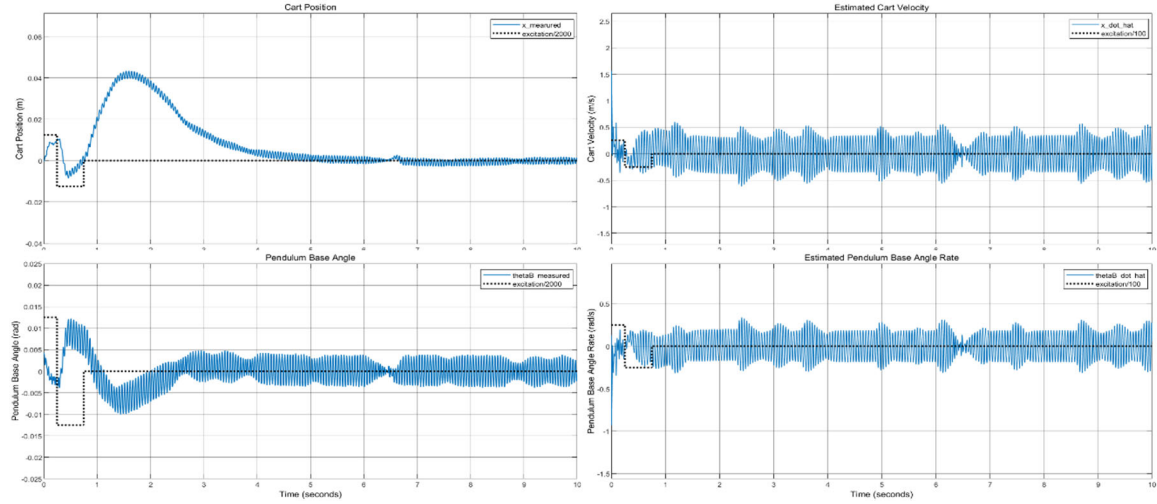


Figure 6.23 Penny FIP states under linear control with flex present.

Given the excessive chatter exhibited in Figure 6.23, state oscillations are certainly feeding back into the controller. It would seem sensible at this point to filter the signals of their high-frequency content before passing them to the controller or filtering the control signal before passing it to the plant. This is not done here, however, as the intent is to excite the system to investigate adaptive augmentation architectures which mitigate this “disturbance.” The adaptive augmentation for output regulation is now reenabled and the adaptive gain retuned ( $\sigma_e = 2200$ ) with comparative results in Figure 6.24.

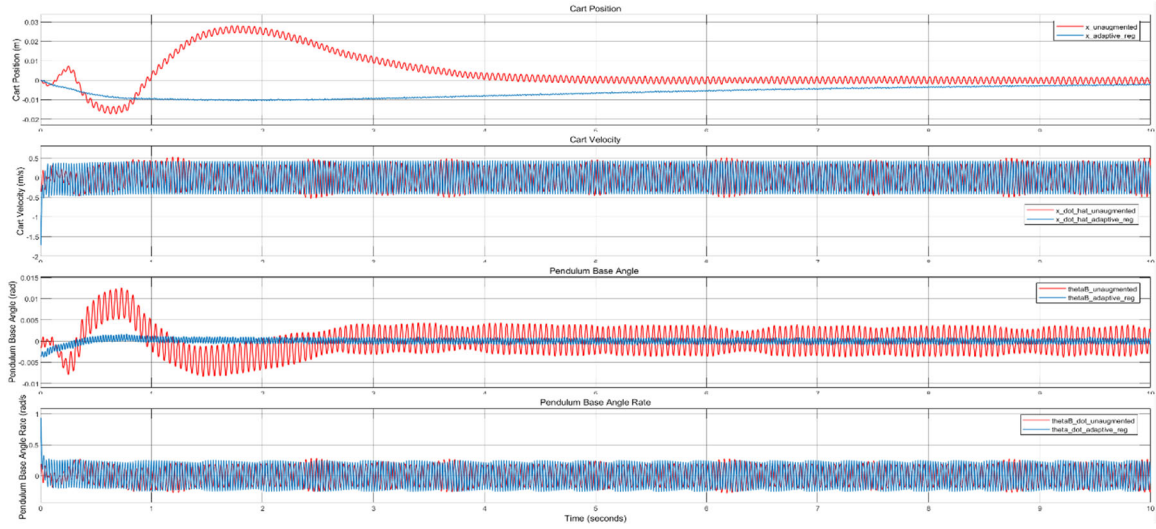


Figure 6.24 Penny FIP augmented with adaptive regulation.

## 6.6. FIP Augmentation with Direct Adaptive Disturbance Accommodation

The adaptive regulator clearly helps, but fast chatter persists and resists all attempts to mitigate through adaptive regulation. We further augment the Penny FIP system with disturbance accommodation, specifically targeting its oscillatory dynamics.

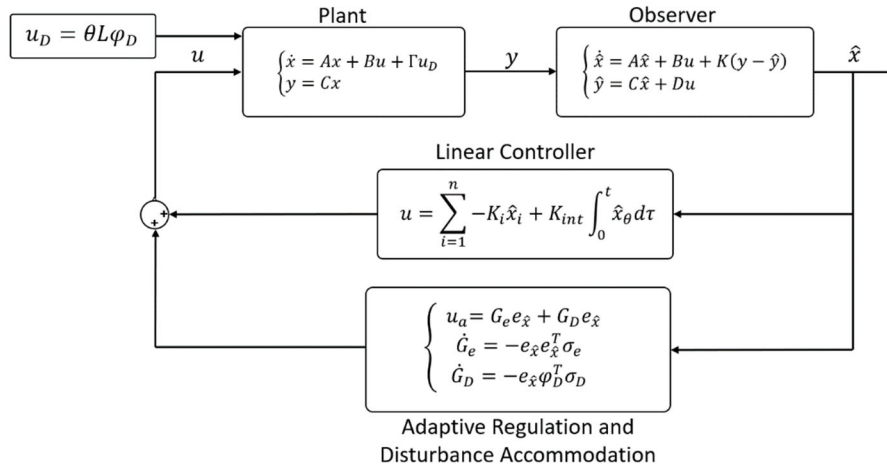


Figure 6.25 Penny FIP augmented with adaptive regulation.

The Simulink UI control model is again updated to implement the adaptive control law, identical in form to Figure 6.12, as shown in Figure 6.26. The operator  $\varphi_D$  is updated with the FIP pendulum basis functions.

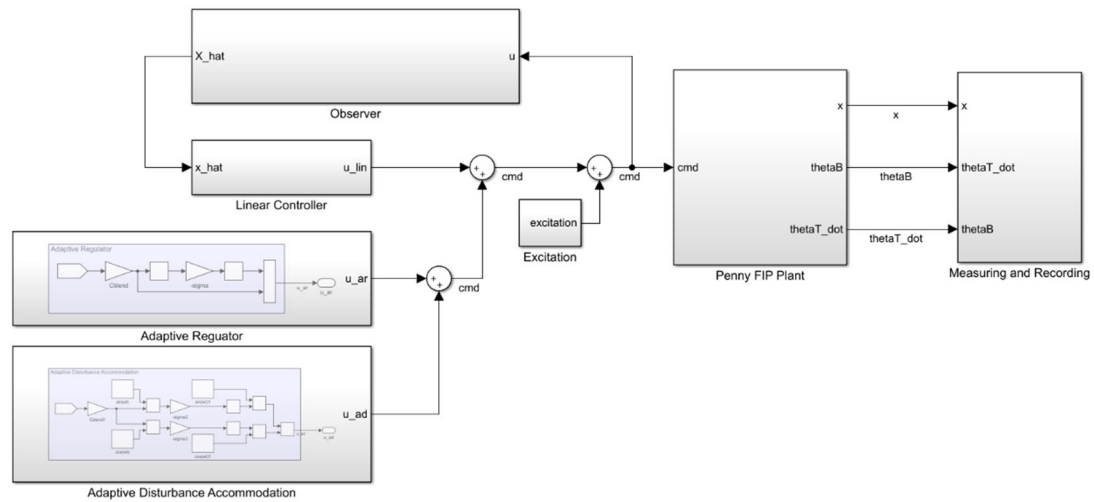


Figure 6.26 Penny FIP with adaptive regulation and disturbance accommodation.

Tuning of the adaptive disturbance gain found that  $\sigma_D = 0.2$  resulted in good performance without introducing instability. Figure 6.27 compares angular states for the pendulum base with and without augmentation. A full state comparison is presented in Figure 6.28.

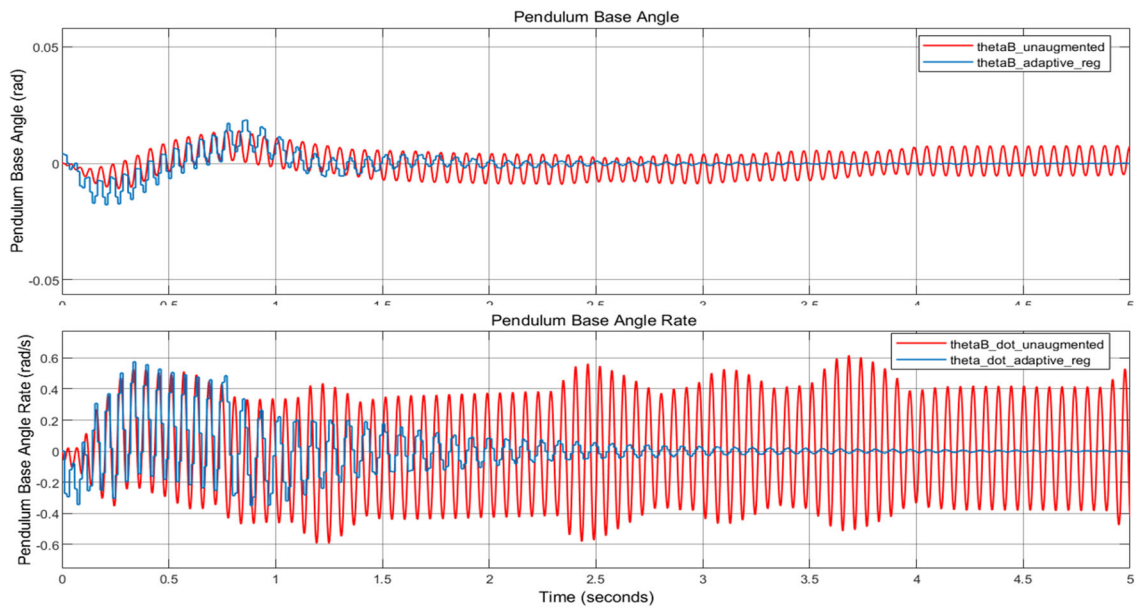


Figure 6.27 Penny FIP angular state comparison with and without augmentation.

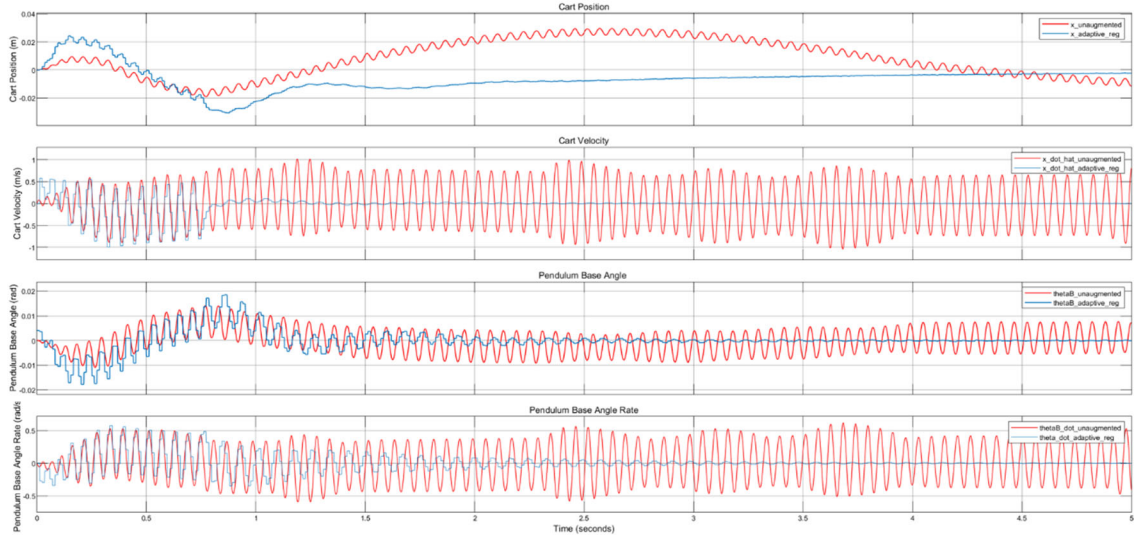


Figure 6.28 Penny FIP with adaptive augmentation.

### 6.7. FIP Augmentation with Disturbance State Adaptive Accommodation

Next, we investigate mitigation of the disturbance state ( $\delta_{\theta}$ ) described in Section 5.3.5. The state-space model from Equation (5.36) has the input  $u$  to output  $\delta_{\theta}$  transfer function:

$$TF_{u \rightarrow \delta} = \frac{-71s - 237.5}{s^3 + 46.18s^2 + 5272s + 2.201e5} \quad (6.10)$$

with numerator roots  $[-3.345 \ 0]$  (note the  $s$  cancelation in the transfer function).

Sensor blending is again employed to make the system appear ASPR to the adaptive controller, with desired zeros arbitrarily placed at  $[-1 \ -2 \ -10]$ . This results in the blended output:

$$C_{blended} = [0.0097 \ -0.0001 \ -0.1359 \ -0.0141] \quad (6.11)$$

Therefore, the input to the control law shall be  $[0.0097x + 0.0001\dot{x} - 0.1359\delta - 0.0141\dot{\delta}]$  where  $\dot{\delta}$  is simply computed by differentiating and filtering  $\delta$ . The same adaptive augmentation architecture is used from Section 6.5.

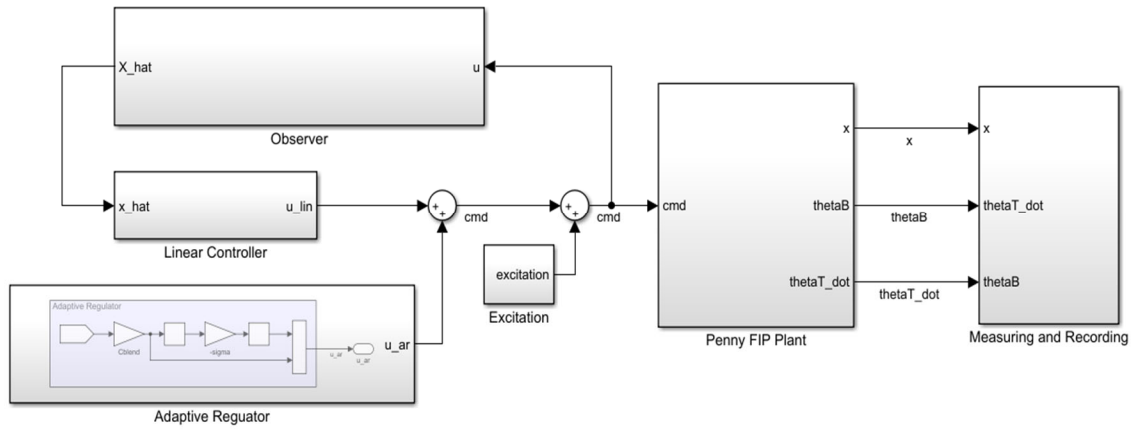


Figure 6.29 Penny FIP augmented with adaptive regulation including disturbance state.

The base state comparison where the adaptive gain  $\sigma = 30$  is shown in Figure 6.30.

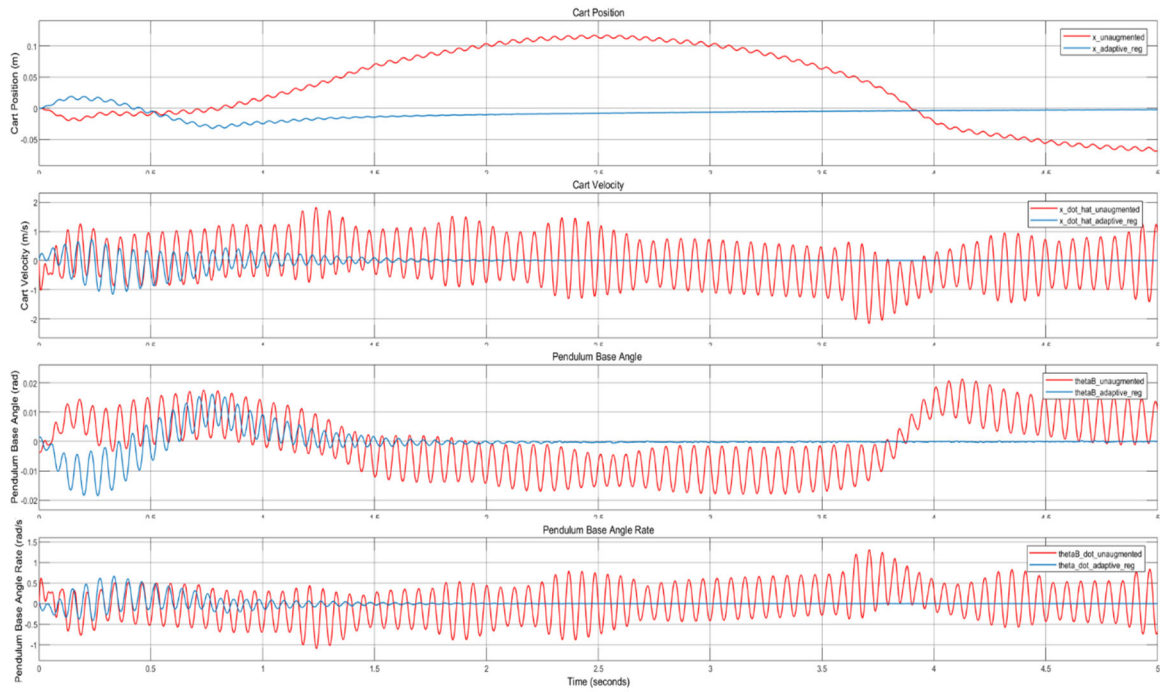


Figure 6.30 Penny FIP augmented vs. unaugmented states,  $\sigma = 30$ .

Figure 6.31 shows the disturbance state comparison for the same test ( $\sigma = 30$ ).

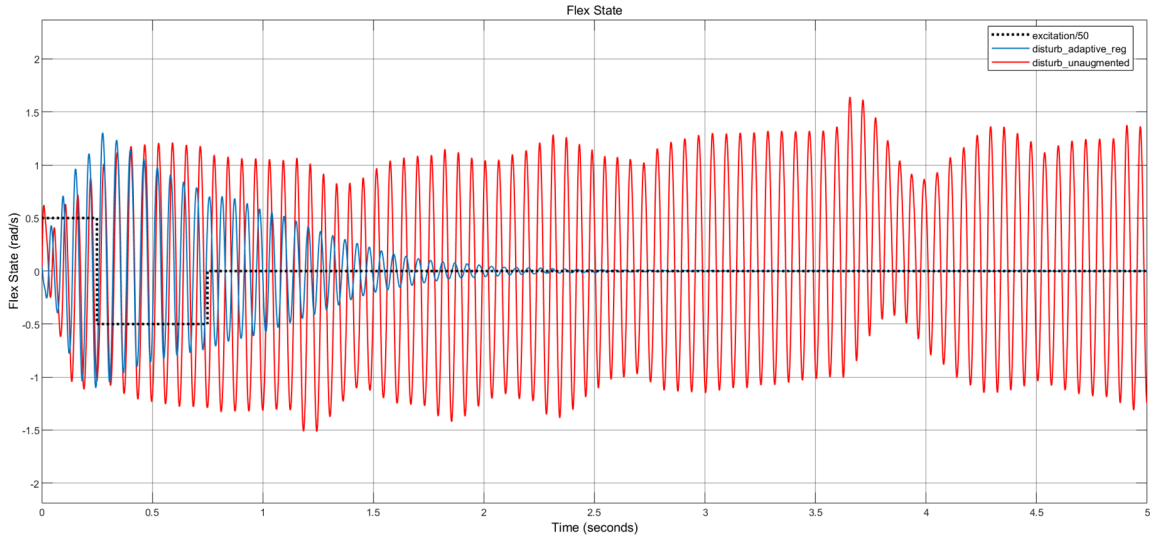


Figure 6.31 Penny FIP augmented vs. unaugmented disturbance states,  $\sigma = 30$ .

### 6.8. FIP Augmentation with Step Disturbance

The purpose of these tests was to investigate the efficacy of the adaptive augmentation in the presence of a step disturbance to determine if adaptive regulation and/or disturbance accommodation yielded benefits over the linear controller alone.

The idea for this test was to apply a step disturbance of increasing amplitude to the unaugmented system until the pendulum angle was no longer recoverable. Next, the same would be performed to the augmented system, as shown in Figure 6.20, and the results compared. It was soon determined that the augmentation made no difference to this critical step amplitude, although it took many tests to learn this, as differing initial conditions make FIP testing exceedingly difficult. The reason the augmentation was of no benefit was because the linear controller had evidently been tuned so aggressively that severe destabilizing events easily saturated the actuators. Of course, when the adaptive

augmentation signal is added to a control signal that is already saturated, no additional actuation effort is produced.

To perform the desired testing, there would need to be remaining control bandwidth for the adaptive augmentation to utilize. To that end, a new experiment was devised. The linear controller's desired poles and integral gain would be successively decremented in 10% increments until the system could not recover from a step amplitude of 50 for 0.5 seconds. Then the control gains would be left at the last recoverable increment and the system would be adaptively augmented as before. Using this process, the last recoverable gains were

$$\begin{cases} K_c = [-5.47 & -152.1 & -421.0 & -161.7] \\ K_i = 900 \end{cases} \quad (6.12)$$

Figure 6.32 shows cart and pendulum base angle states with these updated linear gains. Note the units of pendulum base angle is now in degrees.

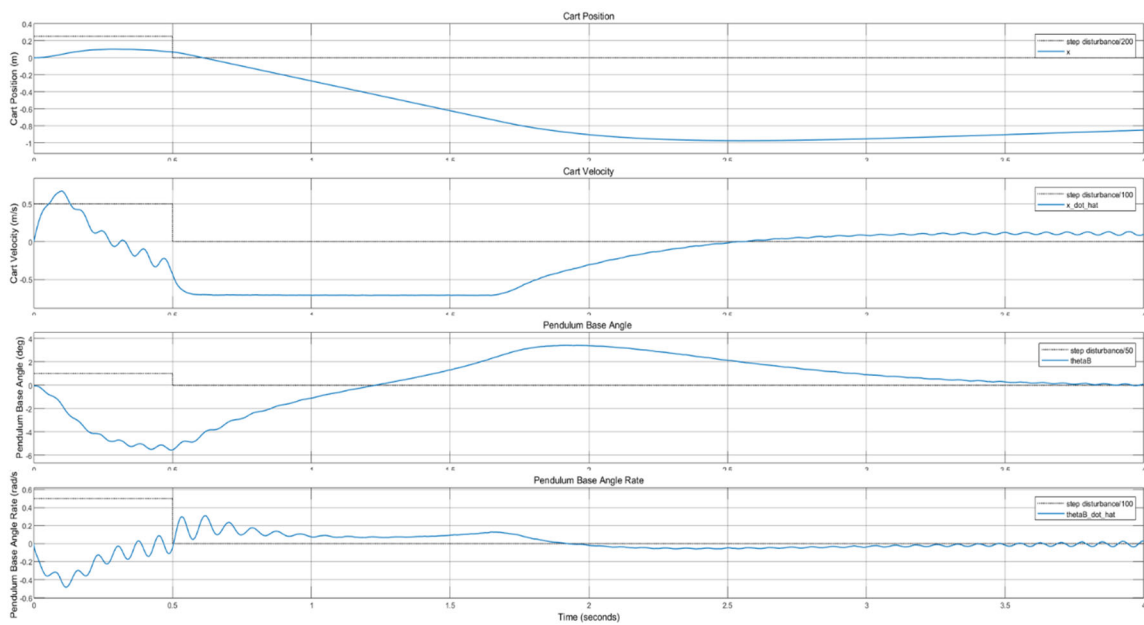


Figure 6.32 – Penny FIP with step disturbance, unaugmented.

Next, the system is augmented with adaptive output regulation as shown in Figure 6.20. The adaptive gain was set at  $\sigma = 5e6$  for good performance, resulting in the plot shown in Figure 6.33 for the same step input (amplitude of 50 for 0.5 seconds). The step disturbance amplitude was increased in increments of approximately 5 until the system could no longer recover pendulum balance. Step amplitudes of 75 were nearly always recoverable (based on initial conditions), and step amplitudes of 79 (the control input limit for Penny) were not recoverable.

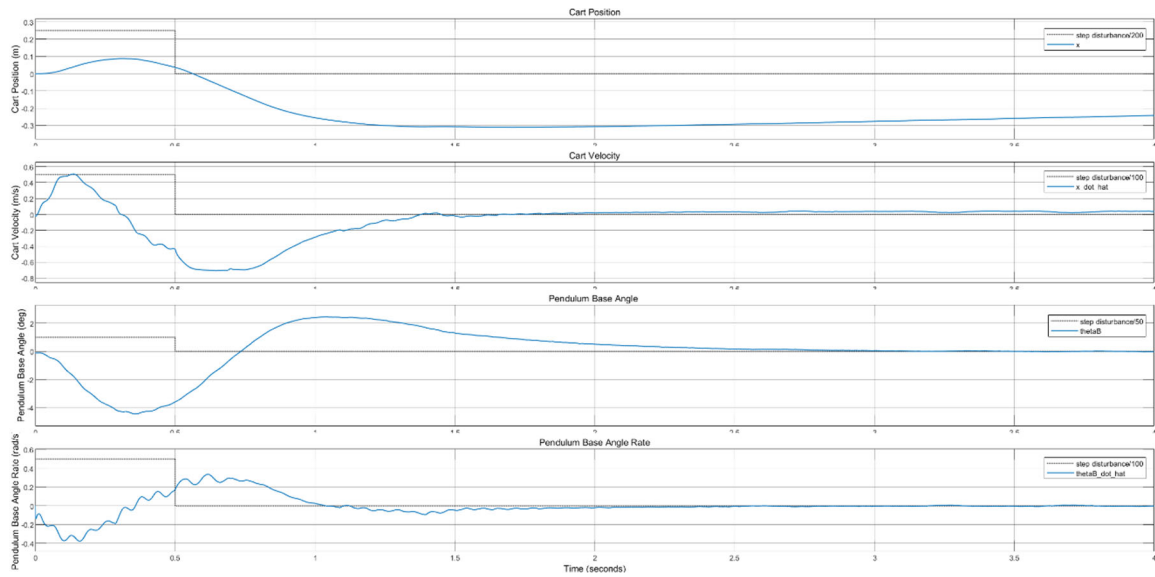


Figure 6.33 FIP with step disturbance augmented with adaptive state regulation.

## 7. Results and Future Work

Under this work, a theoretical foundation for direct adaptive augmentation was presented, followed by a description of the testbed hardware. The software developed to enable hardware functionality was also described, as well as the processes resulting in linear and nonlinear models of the hardware configurations tested herein, CFS and FIP. Finally, each configuration was tested using various control architectures with state responses included. This work assumes a linear controller is managing the system's rigid body states; the cart for CFS, and the cart and rigid pendulum system for FIP.

### 7.1. Penny CFS Implementation

The CFS cart system was first brought under position and velocity control using a separation principle-based controller/observer pair with state responses to a step input shown in Figure 6.3. Since only the cart system is being controlled, the oscillatory dynamics of the beam unfold unabated by the control law (Figure 6.4).

The system was then augmented for adaptive regulation. Since the linear model of the disturbance states  $(\theta, \dot{\theta})$  is not ASPR, sensor blending was used to make the system appear ASPR to the adaptive controller. Figure 6.9 shows the adaptive augmentation rapidly suppressing the flex state.

Finally, CFS was adaptively augmented for disturbance accommodation, where the  $\varphi_D$  term in the adaptive update contains the known basis functions of the disturbance. Sensor blending was again used to make the non-ASPR system appear ASPR, resulting in even more rapid suppression of the disturbance states as shown in Figure 6.13.

Note that in all cases the linear controller is not operating on disturbance state information. There is no doubt a linear controller could be added to manage disturbance

states, which is an area for future work. The current study seeks only to effectively apply adaptive augmentation on hardware to validate the approach.

## 7.2. Penny FIP Implementation

The FIP rigid (braces on) cart-pole system was first brought under position  $(x, \theta)$  and velocity  $(\dot{x}, \dot{\theta})$  control using a separation principle-based controller/observer pair.

Unfortunately, no amount of controller tuning would balance the rigid inverted pendulum for any sustained period, necessitating the addition of integral control. This resulted in a closed-loop stable system with the state responses to a doublet input shown in Figure 6.18.

The FIP rigid body system was then adaptively augmented with state regulation and tested with the same doublet input used in the previous test. As with CFS, the rigid FIP transfer function from input to theta states is not ASPR, so sensor blending was again used to make the system appear ASPR to the adaptive controller. Figure 6.22 shows substantial improvement in angle and angular rate convergence. Note that the disturbance accommodating adaptive augmentation is not added to the rigid system since the pendulum flex is considered the disturbance state here, and with the braces on, no flex is measured.

The pendulum braces were then removed, the adaptive regulator disabled, and the linear controller alone was allowed to balance the FIP system during and after a doublet excitation. This it did, but not without a great deal of fast oscillation caused by the system's flexible modes, seen in Figure 6.23.

The adaptive regulator was reenabled and the doublet test was again performed with the augmentation acting on the previous blended state. Figure 6.24 shows significant

state regulation improvement over linear control alone, although the fast oscillations remained.

Next, the augmented FIP system was additionally augmented with disturbance accommodation of the form shown in Figure 6.25, where the  $\varphi_D$  term in the adaptive update contains the known basis functions of the disturbance. As before, the system's u-to-theta transfer function is not ASPR, so sensor blending was again employed. Figure 6.27, which presents the angular state responses of this control architecture during and after a doublet excitation, show good suppression of the flex motion and full attenuation in approximately 5 seconds.

The next test utilizes the augmented adaptive regulation architecture again, although this time using the 4-state system  $[x \ \dot{x} \ \delta \ \dot{\delta}]^T$  where  $\delta$  is a disturbance state defined as  $\delta = \dot{\theta}_{base} - \dot{\theta}_{tip}$ . Employing augmented adaptive state regulation on this system, again using sensor blending, resulted oscillation suppression in under 3 seconds when excited by the same doublet as the previous test (see Figure 6.31).

Finally, the efficacy of adaptive augmentation was tested in the case of a step disturbance to see if the adaptive augmentation improved controller performance for dispersions far from equilibrium. Here it was noted that the linear controller quickly saturated the actuators, leaving no actuation bandwidth for the adaptive controller to use. The linear controller gains were therefore reduced to the lowest settings that still balanced the pendulum when faced with an amplitude 50 disturbance for 0.5 seconds. The system was then augmented for adaptive regulation with tuned static gain, resulting in recovery from disturbances of step amplitude 75 for 0.5 seconds. This test showed that, if control bandwidth remains, the system's augmentation by an adaptive regulator of

the form shown in Figure 6.20 may help with system recovery when faced with disturbance causing large dispersions.

It should be noted that, for all of the adaptive augmentations discussed in this section, a wide range of gains ( $\sigma$ ) resulted in satisfactory controller performance. The figures presented in Section 6 represent settings which yielded the good performance over many test runs. Also, when an adaptive augmentation was compared to a nonadaptive controller, an effort was made to compare plots with similar initial conditions, since these conditions had a major impact on controller performance in all cases.

### **7.3. Future Work**

While applying direct adaptive control augmentation to hardware, several avenues of interesting research have surfaced. A few remaining deficiencies with the testbed hardware (Penny) could also be addressed and are discussed below.

#### **7.3.1. Sensor Blending**

Adaptive augmentation has proven to be a powerful tool to aid a linear controller in managing unmodeled plant dynamics. The ASPR equivalency is also a powerful result, but not a panacea. One benefit of direct adaptive control approaches is that little plant knowledge is required. However, in practice, when aa system is made ASPR through sensor blending, estimation of the unknown states was required, which required a plant model in the observer. An interesting line of investigation would be a formal approach to minimal blending, perhaps alleviating the need for some state estimation.

Along those same lines, throughout this research, the effect of zero-placement was never intuitive, resulting in much trial and error to implement on hardware. The formulations outlined in Section 2.2 guarantee asymptotic state convergence with

bounded adaptive gains, but they do not speak directly to how transmission zeros should be optimally placed.

### **7.3.2. Output Disturbance Accommodation**

In the early stages of the inverted pendulum robot's development, it was noted that the pendulum system's base angle rotary encoder was prone to injecting output disturbances (structured noise) into the system. These output disturbances would be erroneously acted upon by the controller, ultimately destabilizing the system. This issue was attacked on two fronts. First, an IMU was added to the system to provide a base angle measurement. This had the dual benefit of dealing with any angular offset in the system upon startup, since the IMU references the local gravity vector. The second approach was to develop an adaptive augmentation strategy to mitigate output disturbances within the controller/estimator. The addition of the base IMU fixed the output disturbance problem, so adaptive augmentation for output disturbance accommodation was never fully developed and implemented. This is an interesting application of direct adaptive control theory that will be continued.

### **7.3.3. Comparison with nonadaptive augmentation**

This study examined the efficacy of adaptive augmentation on systems already under linear control. Performance gains were indeed evident throughout Section 6, when adaptive augmentation was employed. However, this study has not examined whether expanding the linear controller to operate on the disturbance state would result in similar performance gains. Investigation the efficacy of controller type for flex disturbance accommodation through augmentation would be an interesting and instructive future line of inquiry.

#### 7.3.4. Penny Hardware/Software

Developing comparative data sets was extremely difficult and time consuming. Since there is currently only one robot, augmented versus unaugmented tests are run independently. Also, each plot produced is unique, especially for Penny FIP, because the initial conditions the controller is presented with have a major impact on how states evolve. The pendulum reset servos were one attempt to normalize test runs. Before the servos, the user would need to balance the pendulum manually before the controller engaged. The software compilation time can be a few minutes between runs, so this was an arduous task. But even with the pendulum reset servos, small angular errors and small momentum transfers still vary between runs. One approach might be to code a test orchestrator that looks for matching state conditions before enabling a test run. Also, lining up data in time proved extremely arduous, even when managing the data files in Matlab. Additional work is merited in the area of software automation.

One of Penny's previously mentioned limitations is the data acquisition system's inability to record electrical energy usage data. Sometimes the discriminator between control approaches is the effort expended by each, so that information would make Penny a better research tool.

Along the lines of hardware upgrades, probably the most effective upgrade Penny could receive is a more capable central processor. Computation and I/O bandwidth are very limited with the Arduino Mega, and as testing progresses and became more sophisticated on Penny, the controller clock cycle was periodically reduced to maintain clock rates, which began at 200 Hz but are now 84 Hz for FIP testing that included the tip sensor measurement.

Another useful upgrade would be the drive train actuators. Having actuators with faster response times and less lash in the gear train would reduce some of the remaining system nonlinearities. Actuators which produced more torque over a wider rpm range would also help them not to saturate so quickly. Finally, the constant deadband and operational range adjustments between ESCs made testing more difficult than necessary. Higher fidelity ESCs with better limit fixing capability, along with the ability to tie all wheel rotations together (e.g., with timing belts) would go a long way toward alleviating these difficulties.

## REFERENCES

- Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, pp. vol. 9, No. 3, pp. 31-37.
- Åström, K. J., & Furuta, K. (2000). Swinging up a pendulum by energy control. *Automatica*, 36:287–295.
- Åström, K., & Wittenmark, B. (1995). *Adaptive Control*. New York: Addison-Wesley.
- Balas, M. (1982). Trends in Large Space Structure Control Theory: Fondest Hopes, Wildest Dreams. *IEEE Trans Automatic Control*, AC-27, No. 3.
- Balas, M. (1995). Finite-dimensional direct adaptive control for discrete-time infinite-dimensional linear systems. *Journal of Mathematical Analysis and Applications*, 196(1): pp. 153-171.
- Balas, M. J., & Frost, S. A. (2016). Direct adaptive control for persistent disturbance rejection in linear infinite dimensional systems. *American Control Conference (ACC)*, pp. 2542-2547.
- Balas, M., & Frost, S. (2018). Stabilization of Fixed Gain Controlled Infinite Dimensional Systems with Actuator Dynamics by Augmentation with Direct Adaptive Control. *AIAA Guidance, Navigation, and Control*. Kissimmee.
- Balas, M., & Frost, S. (2019). Stabilization of Fixed Gain Controlled Nonminimum Phase Infinite Dimensional Systems with Actuator Dynamics by Augmentation with Direct Adaptive Control. *AIAA SciTech Forum*. San Diego.
- Balas, M., & Fuentes, R. (2004). A Non-orthogonal Projection Approach to Characterization of Almost Positive Real Systems with an Application to Adaptive Control. *Proceedings of American Control Conference*. Boston.
- Balas, M., Erwin, R. S., & Fuentes, R. (2000). Adaptive control of persistent disturbances for aerospace structures. *AIAA GNC*. Denver.
- Barkana, I. (1983). *Direct Multivariable Adaptive Control with Application to Large Structural Systems*. Troy: Ph.D. Dissertation, ECSE Dept., Rensselaer Polytechnic Institute.
- Barkana, I. (2013). Simple Adaptive Control - A Stable Direct Model Reference Adaptive Control Methodology - A Brief Survey. *Adaptive Control and Signal Processing*.
- Barkana, I., & Kaufman, H. (1985). Global Stability and Performance of an Adaptive Control Algorithm. *International Journal of Control*, 46(6), pp. 1491–1505.

- Barkana, I., Kaufman, H., & Balas, M. (1983). Model reference adaptive control of large structural systems. *AIAA Journal of Guidance*, 6(2), 112-118.
- Boubaker, O. (May 2013). The Inverted Pendulum Benchmark in Nonlinear Control Theory: A Survey. *International Journal of Advanced Robotic Systems*, doi:10.5772/55058.
- Boubaker, O., & Iriarte, R. (2017). *The Inverted Pendulum in Control Theory and Robotics: From theory to new innovations*. IET Control, Robotics and Sensors Series.
- Broussard, J., & Berry, P. (1978). Command Generator Tracking - The Continuous Time Case. *TASC, Tech. Rep*, TIM-612-1.
- Buhler, C. R., Johansen, M., DuPuis, M., Hogue, M., Phillips, J., Malissa, J., & Calle, C. (2020). Current State of the Electrodynamical Dust Shield for Mitigation. *The Impact of Lunar Dust on Human Exploration*. Houston: NASA.
- Cloud, J., Nieves, R., Duke, A., Muller, T., Janmohamed, N., Buckles, B., & DuPuis, M. A. (November 2021). Towards Autonomous Lunar Resource Excavation via Reinforcement Learning. *ASCEND 2021*. AIAA 2021-4217.
- Feuer, A., & Morse, A. (1978). Adaptive Control of Single-Input, Single-Output Linear Systems. *IEEE Trans. Automatic Control*, AC-23, pp. 557-569.
- Florian, R. (2005). *Correct equations for the dynamics of the cart-pole*. Romania: Center for Cognitive and Neural Studies.
- Fradkov, A. L. (1976). Quadratic Lyapunov Function in the Adaptive Stabilization Problem of a Linear Dynamic Plant. *Siberian Mathematical Journal*, 2, pp. 341-348.
- Francis, B., & Wonham, W. (1975). The internal model principle for multivariable regulators. *Automatica*, Vol 12(5), pp. 457-465.
- Frost, S. A., Balas, M. J., & Wright, A. D. (2009). Direct adaptive control of a utility-scale wind turbine for speed regulation. *INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL*, 19:pp. 59-71.
- Fuentes, R., & Balas, M. (2002). Robust Model Reference Adaptive Control with Disturbance Rejection. *Proc. ACC*.
- Garcia, D. F., Perez, A. E., Moncayo, H., Rivera, K., Betancur, Y., DuPuis, M., & Mueller, R. P. (2018). Spacecraft Health Monitoring Using a Biomimetic Fault Diagnosis Scheme. *Journal of Aerospace Information Systems*, 15:7; 396-413.

- Garcia, D., Moncayo, H., Perez-Rocha, A., Rivera, K., Dupuis, M., & Mueller, R. P. (January 2017). Spacecraft Health Monitoring Using a Biomimetic Fault Diagnosis Scheme. *AIAA Information Systems-AIAA Infotech*. AIAA 2017-1294.
- Greene, M., DuPuis, M., Cloud, J., & Dixon, W. E. (January 2021). Simultaneous Trajectory Tracking Control and Online Mass Estimation for a Regolith Excavating Robot via Integral Concurrent Learning. *AIAA Scitech 2021 Forum*. AIAA 2021-1131.
- Greensite, A. (1967). *Analysis and Design of Space Vehicle Flight Control Systems, Volume I - Short Period Dynamics*. NASA CR-820.
- Hartman, A. L. (2011). *Adaptive Control of Nonminimum Phase Systems Using Sensor Blending*. Thesis: University of Wyoming.
- Holzhüter, T. (2004). Optimal regulator for the inverted pendulum via Euler-Lagrange backward integration. *Automatica*, vol. 40, No. 9, pp. 1613-1620.
- Hu, H. (2004). Modeling and Vibration Control of a Flexible Structure Using Linearized Piezoceramic Actuators. *International Conference on Intelligent Mechatronics and Automation*. Chengdu.
- Ioannou, P. A., & Tao, G. (1987). Frequency domain conditions for strictly positive real functions. *IEEE Transactions on Automatic Control*, 32, 53–54.
- Ioannou, P., & Sun, J. (1996). *Robust Adaptive Control*. Prentice Hall, Inc. .
- Jang, J., Alaniz, A., Hall, R., Bedrossian, N., Hall, C., & Jackson, M. (August 2011). Design of Launch Vehicle Flight Control Systems Using Ascent Vehicle Stability Analysis Tool. *AIAA 2011-6652*. AIAA Guidance, Navigation, and Control Conference.
- Janmohamed, N., Cloud, J., Leucht, K., Bell, E., Buckles, B., & DuPuis, M. (November 2021). Mass Inferencing Model Creation and Deployment to the RASSOR Lunar Excavation Robot. *ASCEND 2021*. AIAA 2021-4216.
- Johansen, M., Dupuis, M., III, J. P., Malissa, J., Wang, J., Hogue, M., & Calle, C. (October 2019). Electrodynamic Dust Shield Testing on the Materials on International Space Station Experiment 11. *70th International Astronautical Congress*. Washington, DC.
- Johnson, C. (1976). *Theory of disturbance-accommodating controllers*. (C. T. Leondes, Ed.) New York: Academic Press.
- Kalman, R. (1964). When is a Linear System Optimal? *Transactions of ASME, Journal of Basic Engineering*, Series D 86, 81–90.

- Katayama, S., Yubai, K., & Hirai, J. (2009). Iterative Design of the Reduced-Order Weight and Controller for the H-infinity Loop-Shaping Method Under Open-Loop Magnitude Constraints for SISO Systems. *IEEE Transactions on Industrial Electronics*, pp. 3854-3863.
- Kaufman, H., Balas, M., Bar-Kana, I., & Rao, L. (1981). Model Reference Adaptive Control of Large Scale Systems. *Proceedings 20th IEEE Conference on Decision and Control*, (pp. pp. 984–989.). San Diego, California.
- Kimura, H. (1975). Pole Assignment by Gain Output Feedback. *IEEE Trans. Automatic Control*, 509-516.
- Kitchen-McKinley, S. J., Drakunov, S. V., Mueller, R. P., & Dupuis, M. A. (2016). Modeling and Control of Cold Gas Propulsion for Spacecraft Attitude Control. *Earth and Space 2016 : Engineering for Extreme Environments*.
- Landau, I. D. (1979). *Adaptive Control - The Model Reference Approach*. New York.
- Lundberg, K., & Barton, T. (2010). History of Inverted-Pendulum Systems. *IFAC Proceedings Volumes*, Volume 42, Issue 24,.
- Lyapunov, A. M. (1892). *The General Problem of the Stability of Motion, Doctoral dissertation*. Univ. Kharkov.
- Magni, L., & Scattolini, R. (2004). Stabilizing model predictive control of nonlinear continuous time systems. pp. vol. 28, pp. 1- 11.
- Magni, L., & Scattolini, R. (2004). Stabilizing model predictive control of nonlinear continuous time systems. *Annual Reviews in Control*, vol. 28, No. 1, pp. 1-11.
- Magni, L., Scattolini, R., & Åström, K. (2002). Global stabilization of the inverted pendulum using model predictive control. *Proceedings of the 15th IFAC World Congress*. Barcelona.
- Mazenc, F., & Bowong, S. (2003). Tracking trajectories of the cart-pendulum system. *Automatica*, vol. 39, No. 4, pp. 677-684.
- Mazenc, F., & Praly, L. (2000). Asymptotic tracking of a reference state for systems with a feed-forward structure. *Automatica*,, vol. 36, No. 2, pp. 179-187.
- Monopoli, R. V. (1974). Model Reference Adaptive Control with an Augmented Error Signal. *IEEE Transactions on Automatic Control* , AC-19(5), pp. 474–484.
- Narendra, K. S., & Valavani, L. (1978). Adaptive Controller Design - Direct Control. *IEEE Transactions on Automatic Control* , AC-23, 570–583.

- NASA Image and Video Library*. (2011, May 30). Retrieved from images.nasa.gov:  
<https://images.nasa.gov/details-s134e010144>
- NESC. (2016). *Stability of the Space Launch System (SLS) Flight Control System (FCS) with Adaptive Augmentation*. National Aeronautics and Space Administration.
- Ogata, K. (2002). *Modern Control Engineering* ( 4th Edition ed.). New Jersey: Prentice Hall.
- Orr, J. (2011). *Elastic Model Transitions Using Quadratic Inequality Constrained Least Squares*. NASA Marshall Space Flight Center, Control Systems Design and Analysis Branch .
- Orr, J. S., & VanZwieten, T. S. (2012). Robust, Practical Adaptive Control for Launch Vehicles. *AIAA Guidance Navigation and Control Conference*. Hampton: NASA/Langley Research Center.
- Osborn, P. V., Whitaker, H. P., & Kezer, A. (1961). New developments in the design of Model reference Adaptive Control Systems. *Inst. Aeronautical Sciences*, paper 61-39.
- Park, M. S., & Chwa, D. (2009). Swing-up and stabilization control of inverted-pendulum systems via coupled sliding-mode control method. *IEEE Transactions on Industrial Electronics*, vol. 56, No. 9, pp. 3541-3555.
- Pei, J. R. (2018). Demonstration of the Space Launch System Augmenting Adaptive Control Algorithm on Pole-Cart Platform. Kissimmee, FL: 2018 AIAA Guidance, Navigation, and Control Conference.
- Pei, J., & Rothhaar, P. (2018). Demonstration of the Space Launch System Augmenting Adaptive Control Algorithm on Pole-Cart Platform. *Guidance, Navigation, and Control Conference*. Kissimmee, FL: AIAA.
- Perez, A., Moncayo, H., Prazenica, R., Kern, Z., Zacny, K., Mueller, R., . . . DuPuis, M. (2016). Free-Flying Robotic System for Interplanetary Prospecting and In Situ Resource Utilization. *Earth and Space 2016 : Engineering for Extreme Environments*.
- Perez-Rocha, A., Moncayo, H., Prazenica, R., Zacny, K., Mueller, R., DuPuis, M., & Ebert, T. (January 2016). Control Laws Development for a Free-Flying Unmanned Robotic System to Support Interplanetary Bodies Prospecting and Characterization Missions. *AIAA Guidance, Navigation, and Control Conference*. AIAA 2016-0884.

- Popov, V. (1973). *Hyperstability of Control Systems*. Berlin: Springer.
- Popov, V. M. (1962). Absolute Stability of Non- linear Control Systems of Automatic Control. *Automation and Remote Control*, 22.
- Prazenica, R., Kern, Z., John, T., Moncayo, H., Zacny, K., Mueller, R., & DuPuis, T. E. (January 2016). Vision-Aided Navigation for a Free-Flying Unmanned Robotic System to Support Interplanetary Bodies Prospecting and Characterization Missions. *AIAA Guidance, Navigation, and Control Conference*. AIAA 2016-0888.
- Rahman, Z. A., & Isa, A. A. (2010). Modal Identification of Flexible Structures with Applications in Robotic Manipulators. *Proceedings of the IMAC-XXVIII*. Jacksonville, FL.
- Rigatos, G., Siano, P., Abbaszadeh, M., Ademi, S., & Melkikh, A. (2017). Nonlinear H-infinity control for underactuated systems: the Furuta pendulum example. *The International Journal of Dynamics and Control*, pp. 1–13.
- Siuka, A., & Schoberl, M. (2009). Applications of energy based control methods for the inverted pendulum on a cart. *Robotics and Autonomous Systems*, vol. 57, pp. 1012-1017.
- Sobel, K., Kaufman, H., & Mabus, L. (1982). Adaptive Control for a Class of MIMO Systems. *IEEE Transactions on Aerospace*, 18, pp. 576–590.
- Stansbury, R., Towhidnejad, M., Clifford, J., Dop, M., Hoffman, R., Cione, J., . . . DuPuis, M. (June 2012). The Gale UAS for Tropical Cyclone Measurements: An Update and Lessons Learned. *Infotech@Aerospace 2012*. AIAA 2012-2523.
- Stansbury, R., Towhidnejad, M., Demirkiran, I., Clifford, J., Dop, M., Koung, T., . . . DuPuis, M. (March 2011). A P-3 Deployable Unmanned Aircraft for Scientific Measurement of Tropical Cyclones. *AIAA 2011-1421*. Infotech@Aerospace 2011.
- Tang, J., & Ren, G. (2009). Modeling and simulation of a flexible inverted pendulum system. *Tsinghua Science and Technology*, vol. 14, no. S2, pp. 22-26.
- Tobbe, P., Matras, A., & Wilson, H. (2009). Modeling and Simulation of Variable Mass, Flexible Structures. *AIAA Modeling and Simulation*. Chicago: AIAA-2009-6023.
- Vidyasagar, M. (1993). *Nonlinear Systems Analysis (2nd edition)*. New Jersey: Prentice-Hall.
- Wang, H. O., Tanaka, K., & Griffin, M. F. (1996). An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE Transactions on Fuzzy Systems*, vol. 4, No. 1, pp. 14-23.

- Wen, J., & Balas, M. (1989). Finite-dimensional direct adaptive control for discrete-time infinite-dimensional Hilbert space. *Journal of Mathematical Analysis & Applications*, 143, 1–26.
- Whitaker, H. P. (1959). An Adaptive Performance of Aircraft and Spacecraft. *Inst. Aeronautical Sciences*, paper 59-100.
- Y. Xu, M. I., & Furuta, K. (2001). Time optimal swing-up control of single pendulum. *Journal of Dynamic Systems, Measurement and Control*, vol. 123, No. 3, pp. 518-527.
- Yoshikawa, T., Ohta, A., & Kanaoka, K. (2001). State Estimation and Parameter Identification of Flexible Manipulators Based on Visual Sensor and Virtual Joint Model. *IEEE International Conference on Robotics and Automation*. Seoul.
- Zacny, K., Mueller, R. P., Ebert, T., Dupuis, M., Mumm, E., Neal, D., . . . Hedlund, M. (2015). MicroDrill Sample Acquisition System for Small Class Exploration Spacecrafts. *Earth and Space 2014 : Engineering for Extreme Environments*.

## LIST OF PUBLICATIONS

- Buhler, C. R., Johansen, M., DuPuis, M., Hogue, M., Phillips, J., Malissa, J., & Calle, C. (2020). Current State of the Electrodynamic Dust Shield for Mitigation. *The Impact of Lunar Dust on Human Exploration*. Houston: NASA.
- Cloud, J., Nieves, R., Duke, A., Muller, T., Janmohamed, N., Buckles, B., & DuPuis, M. A. (November 2021). Towards Autonomous Lunar Resource Excavation via Reinforcement Learning. *ASCEND 2021*. AIAA 2021-4217.
- Garcia, D. F., Perez, A. E., Moncayo, H., Rivera, K., Betancur, Y., DuPuis, M., & Mueller, R. P. (2018). Spacecraft Health Monitoring Using a Biomimetic Fault Diagnosis Scheme. *Journal of Aerospace Information Systems*, 15:7; 396-413.
- Garcia, D., Moncayo, H., Perez-Rocha, A., Rivera, K., Dupuis, M., & Mueller, R. P. (January 2017). Spacecraft Health Monitoring Using a Biomimetic Fault Diagnosis Scheme. *AIAA Information Systems-AIAA Infotech*. AIAA 2017-1294.
- Greene, M., DuPuis, M., Cloud, J., & Dixon, W. E. (January 2021). Simultaneous Trajectory Tracking Control and Online Mass Estimation for a Regolith Excavating Robot via Integral Concurrent Learning. *AIAA Scitech 2021 Forum*. AIAA 2021-1131.
- Janmohamed, N., Cloud, J., Leucht, K., Bell, E., Buckles, B., & DuPuis, M. (November 2021). Mass Inferencing Model Creation and Deployment to the RASSOR Lunar Excavation Robot. *ASCEND 2021*. AIAA 2021-4216.
- Johansen, M., Dupuis, M., III, J. P., Malissa, J., Wang, J., Hogue, M., & Calle, C. (October 2019). Electrodynamic Dust Shield Testing on the Materials on International Space Station Experiment 11. *70th International Astronautical Congress*. Washington, DC.
- Kitchen-McKinley, S. J., Drakunov, S. V., Mueller, R. P., & Dupuis, M. A. (2016). Modeling and Control of Cold Gas Propulsion for Spacecraft Attitude Control. *Earth and Space 2016 : Engineering for Extreme Environments*.
- Perez, A., Moncayo, H., Prazenica, R., Kern, Z., Zacny, K., Mueller, R., . . . DuPuis, M. (2016). Free-Flying Robotic System for Interplanetary Prospecting and In Situ Resource Utilization. *Earth and Space 2016 : Engineering for Extreme Environments*.
- Perez-Rocha, A., Moncayo, H., Prazenica, R., Zacny, K., Mueller, R., DuPuis, M., & Ebert, T. (January 2016). Control Laws Development for a Free-Flying Unmanned Robotic System to Support Interplanetary Bodies Prospecting and Characterization Missions. *AIAA Guidance, Navigation, and Control Conference*. AIAA 2016-0884.

- Prazenica, R., Kern, Z., John, T., Moncayo, H., Zacny, K., Mueller, R., & DuPuis, T. E. (January 2016). Vision-Aided Navigation for a Free-Flying Unmanned Robotic System to Support Interplanetary Bodies Prospecting and Characterization Missions. *AIAA Guidance, Navigation, and Control Conference*. AIAA 2016-0888.
- Stansbury, R., Towhidnejad, M., Clifford, J., Dop, M., Hoffman, R., Cione, J., . . . DuPuis, M. (June 2012). The Gale UAS for Tropical Cyclone Measurements: An Update and Lessons Learned. *Infotech@Aerospace 2012*. AIAA 2012-2523.
- Stansbury, R., Towhidnejad, M., Demirkiran, I., Clifford, J., Dop, M., Koung, T., . . . DuPuis, M. (March 2011). A P-3 Deployable Unmanned Aircraft for Scientific Measurement of Tropical Cyclones. *AIAA 2011-1421*. Infotech@Aerospace 2011.
- Zacny, K., Mueller, R. P., Ebert, T., Dupuis, M., Mumm, E., Neal, D., . . . Hedlund, M. (2015). MicroDrill Sample Acquisition System for Small Class Exploration Spacecrafts. *Earth and Space 2014 : Engineering for Extreme Environments*.

## APPENDIX A: SUPPORTING MATHEMATICS

### Proof of Theorem 1

The following proof can be found in Appendix A, Addendum I of (NESC, 2016).

The Linear Matching Conditions (2.13) can be rewritten:

$$\begin{cases} AS_1 + BS_2 = S_1L_m + H_1, \text{ where} \\ CS_1 = H_2 \end{cases}$$

$$S_1 \equiv [S_{11}^* \quad S_{12}^* \quad S_{13}^*], S_2 \equiv [S_{21}^* \quad S_{22}^* \quad S_{23}^*], L_m \equiv \begin{bmatrix} A_m & B_m & 0 \\ 0 & F_m & 0 \\ 0 & 0 & F \end{bmatrix}, \text{ and}$$

$$\begin{cases} H_1 \equiv [0 \quad 0 \quad -\Gamma\theta] \\ H_2 \equiv [C_m \quad 0 \quad 0] \end{cases}$$

Now let  $CB > 0$ . Use the coordinate transformation  $W$  from lemma 2 in (Balas & Fuentes, 2004) to put (A, B, C) into “normal form:”

$$\begin{cases} \dot{y} = \bar{A}_{11}y + \bar{A}_{12}z_2 + CBu \\ \dot{z}_2 = \bar{A}_{21}y + \bar{A}_{22}z_2 \end{cases}$$

$$\begin{aligned} i. e. \exists \text{ nonsingular } W \equiv \begin{bmatrix} C \\ W_2^T P_2 \end{bmatrix} \ni WAW^{-1} &\equiv \bar{A} = \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{bmatrix}, WB \\ &= \begin{bmatrix} CB \\ 0 \end{bmatrix} \equiv \bar{B}, \text{ and } CW^{-1} = [I_m \quad 0] \equiv \bar{C} \end{aligned}$$

$$\Rightarrow \begin{cases} \bar{S}_1 L_m = W S_1 L_m = W A W^{-1} W S_1 + W B S_2 - W H_1 = \bar{A} \bar{S}_1 + \bar{B} S_2 - \bar{H}_1 \\ \quad \quad \quad = \bar{A} \bar{S}_1 + \begin{bmatrix} CB \\ 0 \end{bmatrix} S_2 - \bar{H}_1 \\ \\ H_2 = C W^{-1} W S_1 = \bar{C} \bar{S}_1 = [I \quad 0] \bar{S}_1 = \bar{S}_a \end{cases}$$

where  $\bar{S}_1 \equiv W S_1 = \begin{bmatrix} \bar{S}_a \\ \bar{S}_b \end{bmatrix}$ .

$$\Rightarrow \begin{bmatrix} H_2 \\ \bar{S}_b \end{bmatrix} L_m = \bar{A} \begin{bmatrix} H_2 \\ \bar{S}_b \end{bmatrix} + \begin{bmatrix} CB \\ 0 \end{bmatrix} S_2 - \begin{bmatrix} \bar{H}_a \\ \bar{H}_b \end{bmatrix}$$

$$\Rightarrow \begin{cases} S_2 = (CB)^{-1} [H_2 L_m + \bar{H}_a - (\bar{A}_{11} H_2 + \bar{A}_{12} \bar{S}_b)] \\ \bar{S}_b L_m = \bar{A}_{22} \bar{S}_b + (\bar{A}_{21} H_2 - \bar{H}_b) \end{cases}$$

Now, if  $(\bar{A}_{22}, L_m)$  share no eigenvalues, it is well known (Balas M. , 1995) that

we can solve the above for a unique  $\bar{S}_b$ . Then

$$\begin{cases} \bar{S}_1 = \begin{bmatrix} H_2 \\ \bar{S}_b \end{bmatrix}, S_2 = (CB)^{-1} [H_2 L_m + \bar{H}_a - (\bar{A}_{11} H_2 + \bar{A}_{12} \bar{S}_b)], \text{ and} \\ \bar{A}_{22} \bar{S}_b - \bar{S}_b L_m = \bar{H}_b - \bar{A}_{21} H_2 \end{cases}$$

Since  $(\bar{A}_{22}, L_m)$  sharing no eigenvalues is the same as  $A$  sharing no transmission zeros with  $A_m, F_m$ , or  $F$ .

#

### Proof of Convergence

Theorem 3: Suppose the following are true:

- (1)  $u_m(t)$  is bounded (i.e., all eigenvalues of  $F_m$  are in the closed left-half plane and any eigenvalues on the imaginary axis are simple)
- (2) The reference model (2.9) is stable (i.e., all eigenvalues of  $A_m$  are in the open left-half plane)
- (3)  $\varphi_D$  is bounded (i.e., all eigenvalues of  $F$  are in the closed left-half plane and any eigenvalues on the imaginary axis are simple)
- (4)  $(A, B, C)$  is ASPR, i.e.,  $T(s) \equiv C(sI - A)^{-1}B$  is minimum phase with  $CB > 0$

Here we use the following version of Barbalat's Lemma, see (Popov V. , 1973) pp.

210-211:

**Lemma 2:** If  $f(t)$  is a real, differentiable function on  $(0, \infty)$  with  $\lim_{t \rightarrow \infty} f(t)$  finite and  $\frac{df}{dt}$  uniformly continuous, then  $\lim_{t \rightarrow \infty} \frac{df}{dt} = 0$ .

We have already seen that  $\dot{V} \leq 0$ ; therefore  $V(t) - V(0) = \int_0^t \dot{V}(\tau) d\tau \leq 0$  or  $0 \leq V(t) \leq V(0)$  where  $V(0) < \infty$ . Hence  $\lim_{t \rightarrow \infty} V(t)$  is finite. Also,  $\dot{V}(t)$  is bounded because

$$\begin{aligned}
 \dot{V}(t) &= -(e_*^T Q \dot{e}_*) \leq \|e_*\| \|Q\| \|\dot{e}_*\| \\
 &= \|e_*\| \|Q\| \|A_c e_* + B \Delta G \eta\| \\
 &\leq \|e_*\| \|Q\| (\|A_c\| \|e_*\| + \|B\| \|\Delta G\| \|\eta\|)
 \end{aligned}$$

and  $e_*$  and  $\Delta G$  are bounded by previous argument via Lyapunov theory. Also  $\eta$  is bounded since  $u_m$  is bounded,  $A_m$  is stable,  $e_y = Ce_*$  is bounded, and  $\varphi_D$  is bounded. Thus  $\dot{V}(t) = \int_0^t \dot{V}(\tau) d\tau$  is uniformly continuous and Barbalat's Lemma may be applied to yield:

$$0 = \lim_{t \rightarrow \infty} \dot{V}(t) = - \lim_{t \rightarrow \infty} e_*^T Q e_*$$

Since  $Q > 0$ , we have  $e_* \xrightarrow[t \rightarrow \infty]{} 0$  and  $e_y \equiv y - y_m = Ce_* \xrightarrow[t \rightarrow \infty]{} 0$ .

## APPENDIX B: SOFTWARE

### Pro Micro read MPU-6050 and write to Xbee

```
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high

#define INTERRUPT_PIN 5
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is
6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte
from MPU
uint8_t devStatus; // return status after each device
operation (0 = success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is
42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
VectorInt16 gy; // [x, y, z] gyro sensor
measurements

// scaling factor for gyro measurement
float scalingFactor;

//
=====
// === INTERRUPT DETECTION ROUTINE
===
//
=====

volatile bool mpuInterrupt = false; // indicates whether MPU
interrupt pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
```

```

}

//
=====
// ===                      INITIAL SETUP
=====
//
=====

#include <SoftwareSerial.h>

SoftwareSerial XBee(0, 1); // Arduino RX, TX (XBee Dout, Din)

void setup() {
    XBee.begin(38400);

    // join I2C bus (I2Cdev library doesn't do this
    automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        Wire.setClock(400000); // 400kHz I2C clock. Comment this line
        if having compilation difficulties
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    Serial.begin(9600);
    // while (!Serial); // allow serial to open if we want to
    debug setup logic

    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);

    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection
    successful") : F("MPU6050 connection failed"));

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // supply your own gyro offsets here, scaled for min
    sensitivity
    mpu.setXGyroOffset(110);
    // mpu.setYGyroOffset(76);
    // mpu.setZGyroOffset(-85);
    // mpu.setZAccelOffset(1788); // 1688 factory default

```

```

    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
        // Calibration Time: generate offsets and calibrate our
MPU6050
        mpu.CalibrateAccel(6);
        mpu.CalibrateGyro(6);
        mpu.PrintActiveOffsets();
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.print(F("Enabling interrupt detection (Arduino
external interrupt "));
        Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
        Serial.println(F(")..."));
        attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN),
dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop() function
knows it's okay to use it
        Serial.println(F("DMP ready! Waiting for first
interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);

    // set gyro limits
    mpu.setFullScaleGyroRange(MPU6050_GYRO_FS_250);
    Serial.print("Actual MPU range: ");
    Serial.println(mpu.getFullScaleGyroRange());

    // total range of values reported
    int numBins = 2862; // empirically measured by looking at
scope/monitor
    // max dps precision plus minus
    int dpsRange = 250 * (1 + mpu.getFullScaleGyroRange()); //
250, 500, 1000, 2000

```

```

    // degree measurement = relativeMeasurement * scalingFactor
    Serial.print("dps range: ");
    Serial.println(dpsRange);
    scalingFactor = (dpsRange * 2) * 1.0 / numBins; // make sure
this is float
    Serial.print("Scaling factor: ");
    Serial.println(scalingFactor, 4);
}

//
=====
// ===          SCALE GYRO TO BE WITHIN RANGE OF uint8_t
//
=====

void sendScaledGyro(float val) {
    uint8_t scaledGyro;
    float scaledGyro_f = scalingFactor * val;

    if (scaledGyro_f > 127) {
        scaledGyro = 255;
    } else if (scaledGyro_f < -128) {
        scaledGyro = 0;
    } else {
        scaledGyro = round(scaledGyro_f) + 128;
    }

    XBee.write(scaledGyro);
    // Serial.print("Scaled Gyro X\t");
    // Serial.println(scaledGyro); // this is around axis of
pendulum flex
}

//
=====
// ===          MAIN PROGRAM LOOP
//
=====

void loop() {
    // if programming failed, don't try to do anything
    if (!dmpReady)
        return;
    // read a packet from FIFO
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the
Latest packet

        // OUTPUT_READABLE_GYRO
        mpu.dmpGetGyro(&gy, fifoBuffer);
        Serial.print("Raw Gyro XYZ\t");

```

```

        Serial.println(gy.x); // this is around axis of pendulum
flex
        sendScaledGyro(gy.x);

        // blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);
    }
}

```

## Microcontroller 2: Tip rate read/write

```

#include <Wire.h>
const byte numBytes = 15;
byte receivedBytes[numBytes];
float rChunk = 0b0;
float pChunk = 0b0;
float yChunk = 0b0;
byte checksum1 = 0b0;
byte checksum2 = 0b0;
byte numReceived = 0;
float prevChunk = 0b0;
boolean newData = false;

void setup() {
    Serial.begin(115200);
    Serial1.begin(115200);

    Serial.println("<Arduino is ready>");

    pinMode(22, OUTPUT);
    pinMode(23, OUTPUT);
    pinMode(24, OUTPUT);
    pinMode(25, OUTPUT);
    pinMode(26, OUTPUT);
    pinMode(27, OUTPUT);
    pinMode(28, OUTPUT);
    pinMode(29, OUTPUT);
    pinMode(30, OUTPUT);
    pinMode(31, OUTPUT);
    pinMode(32, OUTPUT);
    pinMode(33, OUTPUT);
    pinMode(34, INPUT);
}

void loop() {
    recvBytesWithStartEndMarkers(); // Collect data from microstrain

```

```

    parseDataToChunks(); // Parse the data into each axis chunk and the two checksum
    values
    formatData(); // Prepare and send the data over the 12-wire bundle
    showNewData(); // Print data to connected serial port (optional)
}

void sendData(int num) {
    String in = String(num, BIN); // Convert the data to a binary string
    Serial.println(in);

    String zero = "0";
    while (in.length() < 12) { // Make sure the data is 12 characters long by adding zeros
        to the front
        in = zero + in;
    }
    //Serial.println(in);
    digitalWrite(34, LOW); // Set the pin low so that simulink knows not to read

    // These if statements adjust the pins of the 8-wire bundle to send the data to
    simulink
    if (in.substring(0, 1).equals("1")) {
        digitalWrite(22, HIGH);
    } else {
        digitalWrite(22, LOW);
    }
    if (in.substring(1, 2).equals("1")) {
        digitalWrite(23, HIGH);
    } else {
        digitalWrite(23, LOW);
    }
    if (in.substring(2, 3).equals("1")) {
        digitalWrite(24, HIGH);
    } else {
        digitalWrite(24, LOW);
    }
    if (in.substring(3, 4).equals("1")) {
        digitalWrite(25, HIGH);
    } else {
        digitalWrite(25, LOW);
    }
    if (in.substring(4, 5).equals("1")) {
        digitalWrite(26, HIGH);
    } else {
        digitalWrite(26, LOW);
    }
    if (in.substring(5, 6).equals("1")) {

```

```

    digitalWrite(27, HIGH);
  } else {
    digitalWrite(27, LOW);
  }
  if (in.substring(6, 7).equals("1")) {
    digitalWrite(28, HIGH);
  } else {
    digitalWrite(28, LOW);
  }
  if (in.substring(7, 8).equals("1")) {
    digitalWrite(29, HIGH);
  } else {
    digitalWrite(29, LOW);
  }
  if (in.substring(8, 9).equals("1")) {
    digitalWrite(30, HIGH);
  } else {
    digitalWrite(30, LOW);
  }
  if (in.substring(9, 10).equals("1")) {
    digitalWrite(31, HIGH);
  } else {
    digitalWrite(31, LOW);
  }
  if (in.substring(10, 11).equals("1")) {
    digitalWrite(32, HIGH);
  } else {
    digitalWrite(32, LOW);
  }
  if (in.substring(11, 12).equals("1")) {
    digitalWrite(33, HIGH);
  } else {
    digitalWrite(33, LOW);
  }
  digitalWrite(34, HIGH); // Set the pin high to let simulink know that it is ok to read
the data
}

void formatData() {
  if (abs(pChunk - prevChunk) < 1) {
    //Serial.println(pChunk);

    int temp = pChunk * 8900; // We can only send 0-4096 so we multiply the raw
float to set the final value within that range
    if (temp >= 2048) {
      temp = 2047;
    }
  }
}

```

```

    } else if (temp <= -2048) {
        temp = -2047;
    }
    temp += 2048;
    prevChunk = pChunk;
    sendData(temp);
    Serial.println(temp);

} else {
    int temp = prevChunk * 8900;
    if (temp >= 2048) {
        temp = 2047;
    } else if (temp <= -2048) {
        temp = -2047;
    }
    temp += 2048;
    sendData(temp);
}
delay(5);
}

void parseDataToChunks() {
    // Roll Chunk
    uint32_t temp = 0b0;
    for (byte i = 0; i < 4; i++) { //Takes each data value and builds the IEEE floating
        point number from the binary
        temp = temp << 8;
        temp |= receivedBytes[i] & 0xFF;
    }
    rChunk = *(float*)&temp;

    // Pitch Chunk
    temp = 0b0;
    for (byte i = 4; i < 8; i++) {
        temp = temp << 8;
        temp |= receivedBytes[i] & 0xFF;
    }
    pChunk = *(float*)&temp;

    // Yaw Chunk
    temp = 0b0;
    for (byte i = 8; i < 12; i++) {
        temp = temp << 8;
        temp |= receivedBytes[i] & 0xFF;
    }
    yChunk = *(float*)&temp;
}

```

//Checksum, we know the final two bites are the checksum value, so we simply assign them

```

    checksum1 = receivedBytes[13];
    checksum2 = receivedBytes[14];
}
void recvBytesWithStartEndMarkers() {
    static boolean recvInProgress = false;
    static byte ndx = 0;
    byte startMarker = 0x0C; // This is the end of the header
    byte endMarker = 0x75; //This is the start of the header, this way the data we receive
    is only the data and checksum
    byte rb;

```

```

while (Serial1.available() > 0 && newData == false) {
    rb = Serial1.read();

```

```

    if (recvInProgress == true) {
        if (rb != endMarker || ndx < 13) { // Read each byte and save to an array
            receivedBytes[ndx] = rb;
            ndx++;
            if (ndx >= numBytes) {
                ndx = numBytes - 1;
            }
        }
        else {
            receivedBytes[ndx] = '\0'; // terminate the string
            recvInProgress = false;
            numReceived = ndx; // save the number for use when printing
            ndx = 0;

```

```

            newData = true;

```

```

        }
    }

    else if (rb == startMarker) {
        recvInProgress = true;
    }
}

```

```

void showNewData() {
    if (newData == true) {
        // Serial.print(yChunk, 6);
        // Serial.print("\t");
    }
}

```

```
// Serial.print(pChunk, 6);  
// Serial.print("\t");  
// Serial.print(rChunk, 6);  
// Serial.println("\t");  
newData = false;  
}  
}
```

## Sensor Blending Matlab Script

```

%% User Inputs
=====
% Define system state-space matrices (D matrix will be zeros)
% Example:
A = [0 1 0; 2 3 4; 5 6 7];
B=[0;1;1];
C=[1 0 0];

% Define desired Numerator roots in Control Canonical Form (CCF)

des_roots=[-2 -1]; % should be stable and of order n-1 for ASPR

=====

%% Define an all-zero feedthrough matrix D of appropriate size
sa=size(A);
sa_rows=sa(1,1);
sa_cols=sa(1,2);
sb=size(B);
sc=size(C);
sc_rows=sc(1,1);
dr=length(des_roots);

D=zeros(sc(1,1),sb(1,2));

%% Define state space and transfer function models for user inspection
sys1=ss(A,B,C,D);
tf1 = tf(sys1); % shows num and den in s polynomial form
[n1,d1]=ss2tf(A,B,C,D); % define n1,d1 as a vector of polynomial
coefficients
nroots=roots(n1); % determine tf1 numerator roots (if any)

%% Test Controllability and Observability

H=ctrb(A,B);
Hrank=rank(H) % display rank of ctrb matrix

O=obsv(A,C);
Orank=rank(O) % display rank of obsv matrix

%% Create numerator factors
% This loop creates the factors of the numerator polynomial from
% desired roots. For example, assume desired roots are -1 and -2,
% then r{1}=[1 -root1], r{2}=[1 -root2],
% corresponding to the factors s+1 and s+2.
r={dr}; % initialize the "r" array
for k=1:1:dr
    r{k}=[1 -des_roots(1,k)];
end
clear k

%% Create the desired numerator polynomial
Cc{1}=r{1};

```

```

for j=2:1:dr
    Cc{j}=conv(Cc{j-1},r{j});
end
CCnum=Cc{dr}; % desired numerator polynomial for CCF
clear j

%% Define Acc (A matrix in control canonical form)
Apoly=charpoly(A); % find characteristic polynomial of A
a=zeros(1,length(Apoly)-1); % initialize an all-zero vector of poly
coeff.
% This loop defines the bottom row of the Acc matrix (A in CCF)
for k=1:1:sa_rows
    a(1,k)=-Apoly(1,length(Apoly)+1-k);
end
Aul=zeros(sa_rows-1,1); % Acc's left-most column above the bottom row
Aur=eye(sa_rows-1); % Acc's remaining columns above the bottom row
Acc=[Aul Aur;a]; % build Acc
clear k

%% Define Bcc (B matrix in control canonical form)
Bcc=[zeros(sa(1,1)-1,1);1];

%% Defines Ccc (C matrix in control canonical form)
% Reverses the order of the desired numerator polynomial
% and defines it as Ccc
Ccc=zeros(1,dr+1); % creates an all-zeros vector of the correct length
v=length(CCnum);
for k=1:1:v
    Ccc(1,k) = CCnum(1,length(CCnum)+1-k);
end
clear k

%% Computes the required blended output
Hcc=ctrb(Acc,Bcc); % Define controllability matrix of canonical system
T=Hcc*H^-1; % Finds T that maps CC form back to original model
Ti=inv(T); % Defines T inverse
AccTest=(Ti*Acc*T)-A; % Verify T maps Acc back to A (should be zero
matrix)
BccTest=(Ti*Bcc)-B; % Verify T maps Bcc back to B (should be zero
vector)
Cblend=Ccc*T % Compute and display blended output matrix
% Cblend makes plant look ASPR to adaptive controller
Dcc=D;

%% Define state space and transfer function models using blended output
sys2=ss(A,B,Cblend,D)
tf2 = tf(sys2) % shows num and den in s polynomial form
[n2,d2]=ss2tf(A,B,Cblend,zeros); % define n,d as as a vector of
% polynomial coefficients

% verify the roots of n2 match desired roots
roots2 = roots(n2)
des_roots'

% Test for positive realness (S/B identically one)
CblendB = Cblend*B
save('sensor_blended_states_FIP.mat','Cblend');

```

### Motor Signal Builder Discussion

Penny's motors are controlled on the Simulink side by using the "Standard Servo Write" block. According to the Arduino documentation, this block writes a voltage value to the continuous rotation servo, which in effect specifies a speed: zero being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement. However, deadband (a band of input values where the output is zero) exists in the range around 90 ( $\approx 86-95$ ). Also, beyond certain values on the lower and upper end of the commanded range, the signal drops out entirely. For these reasons, in addition to the desire to use a command signal of the form  $cmd \in [-\gamma, \gamma]$ , a motor signal builder was created.

Initial testing showed commands of 0 were yielding small rotation of the wheels. It was determined the dead-band was offset from its true interval. Experiments were run to determine what a correct command range should be, and the dead-band was determined to be 88-89. The Arduino servo write input range is 0-180. Therefore, inputs of 90 and higher correspond to controller commands  $1+$ , yielding forward rotation. Inputs of 87 and lower correspond to commands  $-1$  and lower, yielding reverse rotation. For no rotation, 89 corresponds to command 0. The commands, which are ultimately produced from the control law, are limited to  $[-79, 79]$  in Simulink.

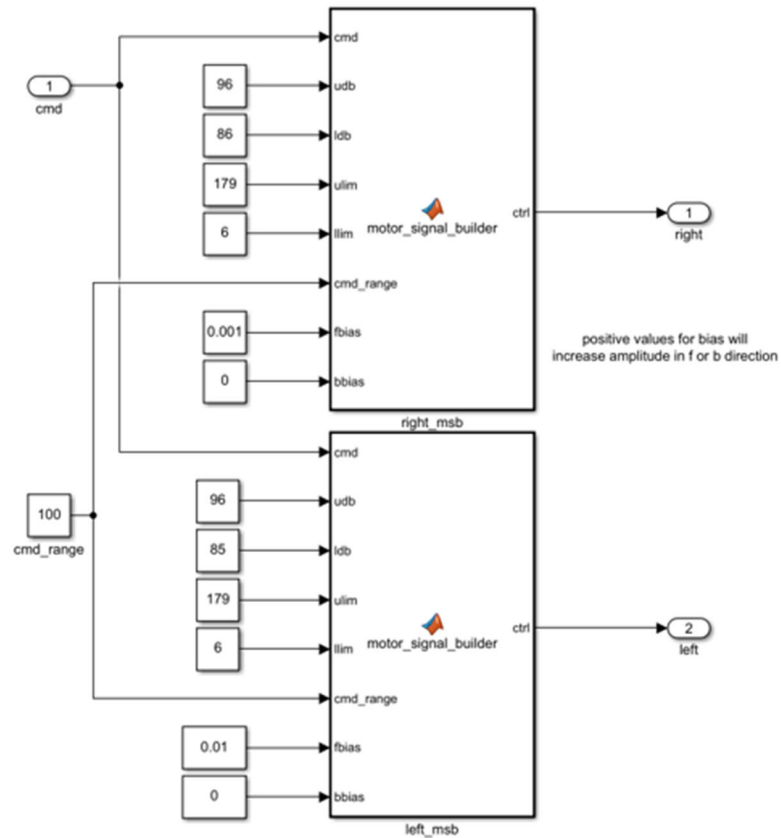
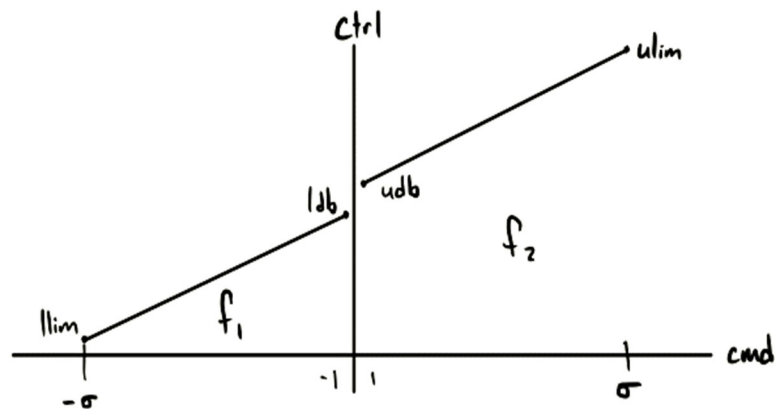


Figure 7.1 Motor Signal Builder

The motor signal builder reduces impact of deadband and nonlinearity in electronic speed controllers on left and right sides of robot.

Figure 7.2 Piecewise function showing mapping  $f: cmd \rightarrow ctrl$

As shown by Figure 7.2,

$$f(cmd) = \begin{cases} llim; cmd \leq -\sigma \\ f_1(cmd); llim \leq cmd < ldb \\ 90; cmd = 0 \\ f_2(cmd); udb < cmd \leq ulim \\ ulim; cmd \geq \sigma \end{cases},$$

the motor signal builder takes  $cmd$  values in the range of  $[-\gamma, \gamma]$  (for *Penny*,  $\gamma = 79$ ) and outputs a  $ctrl$  value in the range of  $[llim, ulim]$  (for *Penny*,  $llim = 6$  and  $ulim = 179$ ). If  $cmd \geq \gamma$ , then  $ctrl = ulim$ ; if  $cmd \leq -\gamma$ , then  $ctrl = llim$ . If  $cmd = 0$ , then  $ctrl = 90$  (i.e. in the deadband which produces no movement). Otherwise, the  $ctrl$  signal is constructed using two piecewise functions, one for each side of the deadband. If  $-\gamma \leq cmd < 0$ , then  $ctrl = f_1(cmd)$ , else if  $0 < cmd \leq \sigma$ , then  $ctrl = f_2(cmd)$ . Two terms  $fbias$  and  $bbias$  are used to change the slope of the linear functions  $f_1$  and  $f_2$  that map  $cmd \rightarrow ctrl$  as shown below.

$$f_1(cmd) = (ldb + cmd) * (1 - bbias)$$

$$f_2(cmd) = (udb + cmd) * (1 + fbias)$$

After creating the  $ctrl$  signal from the input  $cmd$ , it was clear that there were still inconsistencies between the right and left sides of the robot (e.g., one side had a different deadband). Since the left and right sides of the robot are not mechanically linked, each might need slightly different parameters, and so two motor signal builder functions are used, one for each side of the robot.

Given a command  $cmd$  and  $\gamma = 75$ , an output  $ctrl$  is computed according to the following pseudocode:

*If  $cmd \geq \sigma$ , then  $ctrl = ulim$*

*If  $cmd \leq -\sigma$ , then  $ctrl = llim$*

*If  $cmd = 0$ , then  $ctrl = 90$  (i.e. deadband)*

*If  $llim \leq cmd < ldb$ , then  $ctrl = f_1(cmd)$*

*If  $udb < cmd \leq ulim$ , then  $ctrl = f_2(cmd)$*

with  $f_1$  and  $f_2$  defined as follows:

$$f_1(cmd) = (ldb + cmd) * (1 - bbias)$$

$$f_2(cmd) = (udb + cmd) * (1 + fbias)$$

This output command is then sent to the motors.

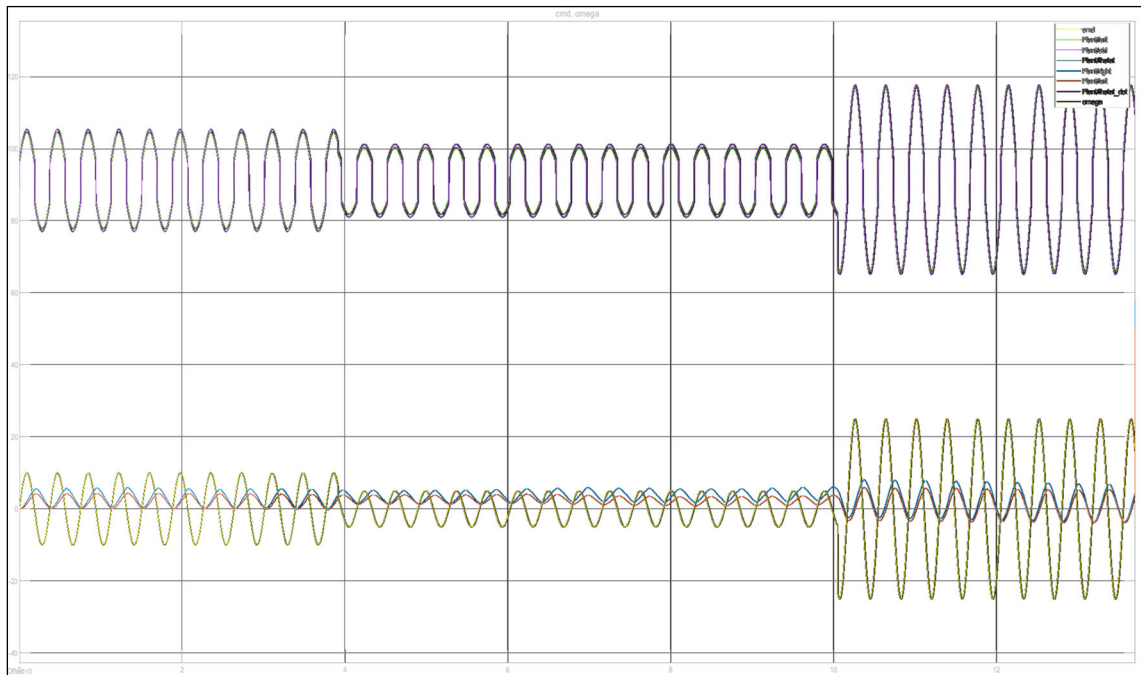


Figure 7.3 Tuned motor signal builder. Yellow:  $cmd$ , purple & green:  $ctrl$ , red & blue: motor encoder readings.

Since the deadband on Penny will occasionally shift, it was desired to automate the process of identifying the deadband. This was accomplished by creating a Simulink model that, given a known starting deadband value  $db$  (such as 90), send a  $ctrl$  value of

$db \pm 1$  (+ if identifying upper deadband, – if identifying lower deadband) and increment/decrement the value of  $db$  until the motor encoders start reporting motion. The first value of  $db$  to produce motion on each side of the robot is then reported to the user using a “Display” block. One consideration made was that small differences in encoder values would falsely trigger the condition indicating that the deadband had been identified; a  $min\_delta$  value was then created, and if  $abs(encoder\_value - prev\_encoder\_value) > min\_delta$ , then the simulation would report the identified deadband.