On Control of Nonlinear Under-Actuated Dynamic Systems (Comparative Study of Modern Control Methods in Application to Swing-up Control of Inverted Pendulum)

by

Dilian Hristov Stoikov
Diploma Engineer, Technical University of Sofia, 2000

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Mechanical Engineering

© Dilian Hristov Stoikov, 2004
University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by
photocopy or other means, without permission of the author.

Supervisor: Dr. Y. Stepanenko and Dr. D. Shpak

Abstract

The thesis presents comparative study of modern control methods for regulating under-actuates systems. The classical pendulum on a cart system, a 2-DOF under-actuated system was utilized as a benchmark system for analyzing controllers' performances.

The work includes analysis, controller design and swing-up control simulation for the following three methods:

- Nonlinear state-space control

  The mathematical model based on the Lie theoretic approach with nonlinear output injection was developed. The controller accomplishes both swing-up and stabilization of the pendulum. The feasibility of the implementation is limited due to physical constraints (length of the cart rail).

- Fuzzy logic control

  A *Sugeno* type fuzzy inference engine implementing the control strategy was built over the system fuzzy model. The simulations showed successful controller behavior but the generated system control input exhibited some non-smoothness that could cause increase on the actuator demand.

- Method of embedded artificial constraints (EAC).

  The analysis reveals dependence between the pendulum motion and the cart acceleration. Using this artificial constraint a state link was developed and the nonlinear control problem was reduced to a linear controller design.

A stabilizing linear state-space controller has been developed and methods for arbitrary pole placement and optimal linear quadratic regulator design were compared. A reduced order current estimator for velocity estimation has been studied and implemented.

A software client/server controller application running on a QNX Neutrino 6.1 platform was developed. The real-time experiments conducted with the EAC/linear state-space controller confirmed good swing-up and stabilization system performance.

Table of Contents

Abstract ii

Table of Contents iv

List of Tables, Figures, and Graphs ix

1 Introduction 1

1.1 Purpose and Scope.....1

1.2 A Benchmark Test System for Under-actuated System Research.....2

1.3 Description of the Benchmark System.....3

1.4 Literature Review.....4

1.4.1 Nonlinear Control.....4

1.4.2 Fuzzy Control.....9

1.4.3 Energy-Based Control.....13

1.4.4 Linear State-Space Control.....15

1.5 Thesis Layout.....17

2 Pendulum Apparatus Specifications Overview 19

2.1 Pendulum on a Cart System Equipment.....20

2.2 PC Based Instrumentation.....21

3 Nonlinear State-Space Controller 23

3.1 Chapter Introduction.....23

3.2 System Dynamic and State-Space representation.....24

3.3 Definitions used in The Nonlinear Control Theory.....27

3.4 Exact Feedback Linearization Method.....29

3.5 Nonlinear Control with Feedback Linearization and Nonlinear Output Injection.....34

3.5.1 Relative System Degree.....34

3.5.2 Linearizations.....38

3.5.3 Controller implementation considerations.....41

3.5.4 Some Comments on the Observer.....43

3.5.5 Simulations.....46

3.5.6 Comments on the Nonlinear State-Space Controller.....49

3.6 Chapter Conclusions.....50

4 Fuzzy Swing-Up Controller **51**

4.1 Chapter Introduction.....51

4.2 Fuzzy Logic Foundations.....52

4.2.1 Membership Functions.....53

4.2.2 Linguistic variables.....53

4.2.3 Fuzzy Controller Design.....54

4.2.3.1 Definitions.....54

4.2.3.2 Controller design stages.....56

4.3 Sugeno Fuzzy Modeling for the Cart Pendulum.....58

4.4 Simulations.....70

4.5 Chapter Concluding Remarks and Recommendations.....73

5 Embedded Artificial Constraints Control.....75

5.1 Chapter Introduction.....75

5.2 System Equations of Motion.....76

5.3 System State-Link Representation.....77

5.4 Artificial Cart-Motion Constraint Development.....78

5.4.1 Pendulum Energy Analysis.....78

5.4.2 Defining Cart Motion Strategy.....84

5.5 Cart Position Controller Design.....90

5.5.1 Cart's Equation of Motion and Transfer Function..... 90

5.5.2 Design Requirements.....92

5.5.2.1 Compensated System Parameter Identification.....93

5.5.2.2 Dominant Pole Locations.....96

5.5.3 Root-Locus Based Design.....97

5.5.4 Frequency Domain Behavior.....103

5.5.5 The Discrete Equivalent.....105

5.5.6 Some Comments on the Compensator's Design.....107

5.6 Swing-Up Controller Implementation.....108

5.7 Simulations.....109

5.8 Experimental results.....117

5.9 Chapter Concluding Remarks.....120

6. Stabilizing Linear State-Space Controller Design & Implementation.....121

6.1 Chapter Introduction.....121

6.2 State Space Modeling for the Pendulum on a Cart System.....122

6.2.1 Linearizations.....123

6.2.2 System Transfer Function.....127

6.3 Controller Design.....128

6.3.1 Controllability.....129

6.3.2 Pole Placement for the Closed –Loop Inverted Pendulum System.....129

6.4 Digital System Model.....132

6.5 State Estimation Issues.....134

6.5.1 System Observability.....135

6.5.2 Full Order Observers.....136

6.5.2.1 Prediction Estimator.....136

6.5.2.2. Full Order Current Estimators.....138

6.5.3 Reduced – Order Predictor Estimators.....140

6.5.4 Integral Observer.....141

6.6 Regulator Design.....145

6.6.1 Experimental Closed-Loop Pole Adjustment.....148

6.7 Concluding Remarks Concerning Linear State-Space Control.....150

7 Software Implementation of the Controller **151**

7.1 Chapter Introduction.....151

7.2 The Operating System Platform.....152

7.3 The Controller Application.....156

7.3.1 The *Server* Application 157

7.3.2 The *Client* Application.....159

7.3 Chapter Conclusions and Recommendations.....160

8 Conclusions **161**

8.1 Control Methods Conclusions.....161

8.1.1 Nonlinear State-Space Controller.....161

8.1.2 Fuzzy Logic and the Pendulum on a Cart System.....162

8.1.3 Embedded Artificial Control (EAC) for the Inverted Pendulum.....163

8.1.4 Stabilizing Linear State-Space Controller.....163

8.2 The Software Implementation.....164

8.3 General Recommendations for Control of Under-actuated Systems.....165

8.4 Contributions.....166

8.5 Areas for Future Study.....166

References **167**

Appendix A The Equations of Motion for the DC Motor.....174

Appendix B Equations of Motion for the Pendulum on a Cart System.....179

Appendix C Physical Values of the System Parameters.....188

Appendix D Matlab Simulation Files.....189

Appendix E Controller C Cource Code.....209

List of Tables, Figures, and Graphs

List of Tables

Table 4.1 Fuzzy Rules.....68

Table 5.1 Lookup table defining values for cart excursion amplitude as a function of θ_{max} (simulations).....110

Table 5.2 Lookup table defining values for cart excursion amplitude as a function of θ_{max} (experiments).....117

List of Figures

Figure 1.1 The Pendulum on a Cart System.....4

Figure 2.1 Pendulum Apparatus Block Diagram.....19

Figure 2.2 Pendulum on a Cart Apparatus.....22

Figure 3.1 Pendulum on Cart system for Nonlinear Control Modeling.....25

Figure 3.2 Exact Feedback Linearization Block Diagram.....33

Figure 3.3 Proposed Nonlinear Controller Block Diagram.....41

Figure. 4.1 Common Membership Functions.....53

Figure 4.2 Pendulum on Cart system for Fuzzy Swing-up Control Modeling.....60

Figure 4.3 Membership Functions for *Pendulum Angle*.....62

Figure 4.4 Membership Functions for *Angular Velocity*.....63

Figure 4.5 Membership Functions for *Cart Position*.....64

Figure 4.6 Membership Functions (singletons) for *Control Signal*.....65

Figure 5.1 Pendulum on Cart system for EAC Modeling.....76

Figure 5.2 State Coupling.....77

Figure 5.3 Dependence Between States Presented by a Link.....78

Figure 5.4 Pendulum on a Cart System Free Body Diagram.....79

Figure 5.5 Components of the Force Fp Acting on the Pendulum Center of Mass.....79

Figure 5.6 Pendulum on a Cart System Free Body Diagram
with Approximated Forces.....80

Figure 5.7 Proposed Cart Motion Strategy.....87

Figure 5.8 Linear Control by Imposing Artificial Constraint Equation.....89

Figure 5.9 Cart System Closed-Loop Block Diagram.....91

Figure 5.10 Controller/Cart system Closed –Loop Block Diagram.....93

Figure 5.11 Cart Motion Scenarios Depending on the Pendulum Swing Amplitude....112

Figure 6.1 Pendulum on Cart system for Linear State-Space Control Modeling.....123

Figure 6.2 Linear State-Space System Closed-Loop Block Diagram.....128

Figure 6.3 Discrete-Time Closed-Loop System Block Diagram.....146

Figure 6.4 Simulative System Response to Stabilizing controller with Integral
Observer.....147

Figure 6.5 Overall Pendulum Swing-up and Stabilization Simulation.....147

Figure 7.1 The QNX Microkernel Provides Full Memory Protection.....153

Figure 7.2 The QNX Neutrino OS as a Team of Processes.....154

Figure 7.3 Multithreaded Client/Server Application Architecture.....157

Figure 8.1 Control Method Grouping.....165

List of Graphs

Graph 3.1 Proposed Nonlinear Controller Simulations.....48

Graph 4.1 Cart/Pendulum System Response to Fuzzy Swing-up Controller and Linear State-Space Stabilizing Controller.....70

Graph 4.2 Pendulum Angle vs. Pendulum Angular Velocity under Fuzzy Control.....71

Graph 4.3 Control Signal Determined by the Fuzzy Engine for the Pendulum Swing-Up and Linear State-Space Control for the Stabilization.....72

Graph 5.1 Work/ Acceleration Ratio vs. Pendulum Angle.....83

Graph 5.2 Cart Motion Over Pendulum Angle Driven Trajectory.....86

Graph 5.3 Cart Velocity with the Cart Following the Prescribed Trajectory.....88

Graph 5.4 Closed-loop Cart System Transient Response.....92

Graph 5.5 Closed-Loop Pole Locations of the Cart System vs. Desired Pole Locations.....97

Graph 5.6 Compensated Cart System Root-Locus 101

Graph 5.7 Compensated Cart System Unit Step Response 102

Graph 5.8 Compensated Cart System Open-Loop Frequency Response.....104

Graph 5.9 Compensated Cart System Closed-Loop Frequency Response.....105

Graph 5.10 Discrete Closed-Loop System Unit Step Response.....106

Graph 5.11 Cart Position (idealized) and Pendulum Angle Response under Energy- Based Controller (simulations).....113

Graph 5.12 Cart Position and Pendulum Angle response under Energy- Based Controller (simulations).....114

Graph 5.13 Pendulum Angle vs. Pendulum Angular Velocity (Energy-Based Controller Simulations).....115

Graph 5.14 Control Signal (Energy-Based Controller Simulations).....116

Graph 5.15 Experimental System Performance.....118

Graph 5.16 Control Signal (Experimental Results).....119

Chapter 1

Introduction

1.1 Purpose and Scope

Control of under-actuated mechanical systems or robots represents an important class of control issues that are becoming increasingly important [3, 58]. Examples of such systems are illustrated in [37, 69]. Frequently cited applications include saving energy and reducing weight; these applications are attained by using fewer actuators. Interest in studying the under-actuated mechanical systems is also motivated by their role as a class of strong, nonlinear systems where complex internal dynamics and difficulties for feedback linearization are often exhibited. Traditional nonlinear control methods cannot address the challenges and new approaches must be developed which will stimulate new results in robot control theory in the near future.

Though the dynamics of the under-actuated mechanical system are well understood, the difficulty of solving the control problem for under-actuated mechanisms is obviously due to reduced input space. Literature on the control of under-actuated systems has recently

been written in [13] and [68], and discussion focuses on two degree of freedom examples [46, 8].

The objective of this thesis is to investigate performance of various modern control methods in application to controlling nonlinear, under-actuated, systems. An existing laboratory pendulum on a cart system provides the test bed for this investigation. Modern control methods that were chosen for simulative comparison for swinging up the pendulum to its unstable equilibrium include nonlinear state space, fuzzy logic, and embedded artificial constraints (EAC) control. To verify the proposed algorithms for swinging-up and stabilization of the pendulum, the EAC swing-up controller is combined with the stabilizing linear state-space controller and real time experiments are conducted on the benchmark pendulum system.

1.2 A Benchmark Test System for Under-actuated System Research

The difficult task of controlling under-actuated dynamic systems is often compounded by the fact that these systems are typically highly nonlinear and complicated. To facilitate experimental research, a test system is required which minimizes the complexity of the regular industrial control applications while still providing suitable under-actuated and nonlinear characteristics. The pendulum on a cart (or cart-pole) system is often

considered to be an accepted test case, or benchmark, for control theory investigation development.

1.3 Description of the Benchmark System

The physical apparatus, known as pendulum on a cart system is simply a pendulum mounted on a trolley. The pendulum mounted on the cart is allowed freely to pivot in a planar motion. The cart motion is restricted to horizontal movement in the plane of the pendulum. A permanent magnet DC motor is usually used to actuate the position of the cart. Thus the cart-pendulum system is a two-degree of freedom system with only one of degree of freedom being directly actuated or the system is a typical *under-actuated system* (Figure 1.1).

The system poses two equilibrium states – a stable equilibrium with the pendulum being at rest at its pendent position and unstable equilibrium with the pendulum being at its inverted vertical position.

The model of the system dynamics is derived and given in the Appendix. The differential equations describing the system behavior contain nonlinear terms which are due to the geometric properties of the system.

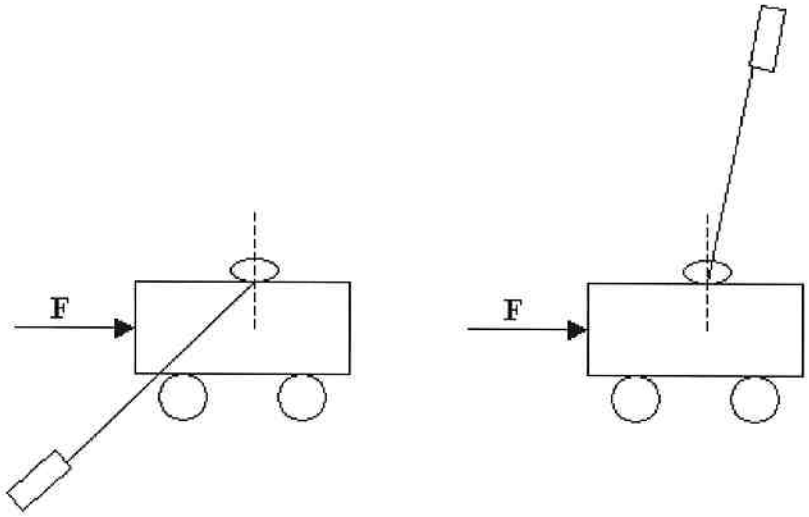


Figure 1.1 The Pendulum on a Cart System

1.4 Literature Review

A literature review and discussion regarding the development of the different modern methods for control and their utilization on the cart-pole system is provided in the next sections.

1.4.1 Nonlinear Control

Early developments in the nonlinear control theory suggest that control of nonlinear

systems be approached by linarizing the system about a nominal point and then applying a feedback law for the resulting linear system. When the system must be controlled over a wide range of operation, this process is repeated at several operating points and a linear controller is designed for the updated linear presentation. The process of switching controllers as the system crosses the specified operation range is referred to as *gain scheduling*. Gain scheduling is often used when controller parameters (feedback gains) are varied according to the system states in a way that pieces together the linear controllers [45]. A variation on gain scheduling is the method of extended linearization. The method allows the feedback gains to vary as a function of the system states, permitting implementation of nonlinear gain specifications. An example of an application of this method is [4]

Another method for controlling a nonlinear system over a wide range of operation is the *exact feedback linearization* method. The theory behind the exact feedback linearization is well studied and defined in [21], [25] and [56]. The basic idea of exact feedback linearization is to construct a nonlinear control law as a so-called inner loop control which, in the ideal case, exactly linearizes the nonlinear system after a suitable state space change of coordinates. The designer can then design a second stage or outer loop control for the linear system presented by the new coordinates.
However, when this approach is applied to under-actuated systems, as is the case of the cart-pendulum system, existence conditions are usually not satisfied.

As opposed to exact feedback linearization, which attempts to transform a nonlinear system to a linear system without any error, *approximate* feedback linearization attempts

to construct a linear approximation about the an equilibrium point which is accurate to a second or higher order. The concept is formulated by Krener in 1984 and further developed in 1987 [29] and [31]. The method considers the nonlinear system

$$\dot{x} = f(x,u) \tag{1.1}$$

which is assumed to be in an equilibrium for $x^0 = 0$ and $u^0 = 0$. The objective is to determine the state and control transformation

$$\begin{aligned} z &= z(\tilde{x}) + O^{p+1}(\tilde{x}) \\ v &= v(\tilde{x},u) + O^{p+1}(\tilde{x},u) \end{aligned} \tag{1.2}$$

(where $\tilde{x} = x - x^0$) which transforms the nonlinear system (1.1) to the order p approximate linear system defined in terms of the new coordinate pair (z, v)

$$\dot{z} = Az + bv + O^{p+1}(\tilde{x},u) \tag{1.3}$$

Krener in [29] formulates the systematic approach for finding the transformations (1.2). Among the benefits of this approach is that the necessary and sufficient conditions are similar to those for exact linearization, yet are much less stringent. This reduces the computational burden for determining existence.

Second-order approximate linearization has been applied to the cart-pendulum problem

by Bedrossian [5]. However, the designed controller is capable of bringing and stabilizing the pendulum at the origin if the initial pendulum angle is not bigger than 60° (with the pendulum angle measured from the inverted position of the pendulum) and thus making the direct use of the method of approximate linearization unsuitable for the pendulum swinging-up problem. A modification of this method that swings-up the pendulum is described in [27] but that approach takes additional assumptions and uses adaptive techniques.

Spong [54] presents the concept *of partial feedback linearization*. The idea is to use nonlinear feedback to linearize the portion of the dynamics corresponding to the actuated (or active) degrees of freedom. The remaining portion of the dynamics after such a partial feedback linearization is nonlinear and represents *internal* or *zero* dynamics of the system. The notion of zero dynamics is discussed in [21] and most generally is described as the internal behavior of the system when input and initial conditions have been chosen in such a way as to constrain the output to remain identically zero. The local stability of the full system may be determined based on the analysis of the two reduced-order systems (the linear system and the system presenting the internal behavior).

Spong refers to this approach as *collocated* input/output linearization. He also describes *non-collocated* input/output linearization, where instead of linearizing the active degrees of freedom, the dynamics associated with the underactuated (passive) degrees of freedom are linearized. This approach, though, results in more complicated analysis regarding both the determination of its existence and stability analysis. Spong gives an example

[54] for applying both the collocated and non-collocated linearization method to the trajectory tracking problem for the *acrobot* system. Unfortunately, as the zero dynamics are determined by the particular control law used and there is no systematic procedure for selecting the feedback control (and more specifically its external component which has to be designed) applying this method to other underactuated systems becomes a rather challenging task.

Variable Structure Control (VSC) provides yet another method for controlling nonlinear systems. VSC is a viable high-speed switching feedback control (for example, the gains in each feedback path switch between two values according to some rules). This variable structure control law provides an effective and robust means of controlling nonlinear plants. It has its roots in relay and bang-bang control theory. Essentially, VSC utilizes a high-speed switching control law to drive the nonlinear plant's state trajectory onto a specified and user-chosen surface in the state space (called the sliding or switching surface), and to maintain the plant's state trajectory on this surface for all subsequent time. This surface is called the switching surface because if the state trajectory of the plant is "above" the surface a control path has one gain and a different gain is used if the trajectory drops "below" the surface. The plant dynamics restricted to this surface represent the controlled system's behavior. By proper design of the sliding surface, VSC attains the conventional goals of control such as stabilization, tracking, regulation, etc.

Takeshi Kawashima [24] proposed a variable structure controller performing both swing-up and stabilization of an inverted pendulum. The controller was developed using the nonlinear plant model and saturation of the control input. Since a design of nonlinear

hyperplane is not established yet, the suggested hyperplane for the controller is designed from the pendulum energy balance and further assumptions are made in order for satisfactory behavior of the controller to be achieved.

The controller presented by Kawashima is further developed by Kwon et. al. [32]. Kwon's controller has the advantage of requiring a cart rail length of 40 cm whereas the Kawashima's controller converges if the rail length is at least 80 cm, but this is achieved at the cost of a higher torque required. This, actually, is the main problem of implementing controllers based on variable structure control methods – they usually impose too high and often even unrealistic requirements to the torque supplier which makes their application not very practical.

1.4.2 Fuzzy Control

The foundations of the fuzzy set theory were first developed by Lotfi Zadeh who in 1965 introduced the concept of the linguistic variable having lexical instead of numeric values. [71]. Zadeh's papers on linguistic fuzzy algorithms [70, 72] motivated research on linguistically conceived control by Mamdani and his colleagues [36, 26]. In 1974 Mamdani demonstrated that a very simple controller, based on Zadeh's fuzzy theory, could regulate a steam engine. In 1980 F.L. Smidth Corp. of Denmark presented a fuzzy controlled cement kiln - the first permanent industrial application. After 1991 fuzzy technology came out of the scientific laboratories and became an industrial tool. Just a small number of the successful projects which at the same time demonstrate the huge diversity of possible application include: automatic control of dam gates for hydroelectric

power plants (Tokyo Electric Power), simplified control of robots (Hirota, Fuji, Toshiba), cruise-control for automobiles (Nissan, Subaru), automatic motor-control for vacuum cleaners with a recognition of the surface and degree of soiling (Matsushita), back-light control for camcorders (Sanyo).

The fuzzy logic approach can be especially suitable for control of nonlinear Lagrangian systems, as it does not rely on any detailed knowledge of the underlying dynamics of the system. If an expert human controller can control the system then his/her operations can be modeled by fuzzy logic and thus a mechanism for automatic control will be provided. Namely this is the most attractive part of the fuzzy control approach - the ability of conceiving linguistic control based on human behavior.

Seiji Yasunobu and Munehito Mori [67] proposed a human strategy based fuzzy swing-up controller for the pendulum on a cart system. The controller adjusts the duration of a constant velocity command on the cart. In order to get the control command, the control rule is formulated as follows. It is assumed that the pendulum has been excited by the velocity command candidates of the cart beforehand. After the pendulum energy is predicted it is evaluated by using a fuzzy set. Then, the command of the cart velocity with the best evaluation is selected as the control command. Simulation and experimental results are provided by the author but the main drawback of the method is that no specific criterion for the pendulum energy estimation is given.

Edgar Sanchez *et al.* [46] proposed a fuzzy Proportional –Derivative (PD) controller for swing-up of a stacked pendulum (often referred as *acrobot*). With reference to the conventional PD control theory the fuzzy PD controller uses the error signal and the rate

of change of the error as inputs to the fuzzy engine. The suggested controller is decomposed with respect to the acrobot's two links and, as inputs for each of the fuzzy controllers are considered, the position error and change of position error for each of the links. The scheme used for deriving the inference scheme is quite elegant. Each of the inputs is presented just by two linguistic values. The same form for the set of membership functions for the two inputs is used, and then a scaling factor is introduced to form the actual membership functions. The input ranges are then decomposed into twenty adjacent input combinations (error/change of error) and then the appropriate fuzzy control is evaluated.

The complicated task of defining the appropriate membership functions and suitable linguistic if-then rules required for the fuzzy controller prevents the universal application of fuzzy control to the general unstable control problem. If the cart-pole example is considered, from modern control analysis, a successful stabilizing controller would use all four state variables as fuzzy inputs. An appropriate fuzzy logic description would provide five (seven would be better) linguistic values for each input. This will result in a fuzzy inference system with total of 5^4=625 (or 7^4=2401) linguistic rules to be defined. Fortunately, it has been shown that a fuzzy inverted pendulum stabilizing controller with relatively satisfactory behavior could be realized without a complete set of rules for all possible state variables. However, the iterative design process for determining the appropriate number of rules and membership function styles can become a tedious task even for the reduced scope of the problem.

An example of fuzzy logic in application to balancing an inverted pendulum on a cart

was first presented by Yamakawa in 1987 [66]. The implementation of the fuzzy controller didn't consider centering of the cart and thus the position of the cart was neglected, leaving the remaining three system states as inputs for the fuzzy engine. Yamakawa's reduced state approach defeats in some respects the generality of the fuzzy logic method. Despite the loss of generality in the implementation of the controller, Yamakawa's choice of state reduction is in the spirit of fuzzy control – tweak and fiddle until the controller works.

Kawaji and Teruyiki Maeda (1991) provided an approach similar to Yamakawa's. However, they used the pendulum angle offset and pendulum angular velocity as their two choices for states subject to fuzzification in order to simulate the stabilization of the system (along with centering of the cart) under fuzzy control [23].

In recent years attention has been devoted to the *Takagi-Sugeno* (*TS*) type fuzzy controllers. The TS fuzzy system model suggests combining the available mathematical description of the process with its linguistic description. The TS system model can be constructed if the local description of the dynamic system to be controlled is available in terms of local linear models

$$\dot{\mathbf{x}}(t) = \mathbf{A}_i\mathbf{x}(t) + \mathbf{B}_i u(t) \tag{1.4}$$

This information is then fused with the available if-then rules where the *i*th rule is of the form

IF $\mathbf{x}(t)$ is F_1^i AND...AND $\mathbf{x}_n(t)$ is F_n^i THEN $\dot{\mathbf{x}}(t) = \mathbf{A}_i x(t) + \mathbf{B}_i u(t)$

where F_j^i, j=1, . ,n is the jth fuzzy set of the ith rule.

In their paper [61] Teixeira and Zak give detailed presentation of existence and derivation of methods for constructing TS fuzzy design models and stabilizing feedback control. Further development and direct implementation of a TS fuzzy controller for the cart-pole stabilization problem is proposed by Song and Smith [49]. The authors discuss the derivation of system specific *optimal control table*, from which the fuzzy rule-base can be extracted. This approach results in a rather efficient fuzzy engine – it consists of just three rules.

The TS based fuzzy control scheme may be compared to the gain scheduling approach - both methods are based on deriving linear system models about a number of operating points. If the appropriate linguistic model of the system is obtained, the TS scheme has the advantage of providing smooth transition between the adjacent linear ranges.

1.4.3 Energy-Based Control

The conceptual frame for applying energy-based methods to control of under-actuated mechanical systems is summarized in Spong's papers in 1996 [52] and in 1997 [55].

The energy based control methods for the pendulum swing-up problem presented in the literature, despite variations in their implementation, are mainly concentrated on defining suitable cart acceleration scheme that would gradually add energy to the pendulum until it reaches its inverted position.

Astrom and Furata [3] present the energy swing-up control for a simple pendulum. Their analysis shows that the problem is completely characterized by two parameters. One is the time scale and the other is the ratio of the maximum acceleration on the pivot and the acceleration of gravity. The authors give a detailed analysis of the dependence of the number of the swings that the pendulum will do until reaching the upright position and the applied acceleration when the pendulum crosses its pendent position.

Quifeng Wei et al. (1995) used the following intuitive approach for increasing the energy of the pendulum. Suppose that the pendulum is at rest at its stable equilibrium point (pendent position). The cart is accelerated with constant acceleration for short time. Then the cart is immediately accelerated in the opposite direction for exactly the same time, the cart returns to its initial state but the pendulum does not. The authors provided elegant proof for this along with an expression for the total energy of the pendulum at the moment the cart returns to its initial state. Consequently they proved that that with such a back and forth motion energy is added to the pendulum and it gradually approaches its inverted position [64].

Yoshida (1999) used Furata's analysis for the pendulum energy and derived swing-up controller using a sinusoidal reference for the cart position generated by the pendulum trajectory [69].

Absil and Sepulchre (2001) proposed to swing-up the pendulum by an energy-based approach while making sure that the cart stays sufficiently close to the $x = \dot{x} = 0$ state

(with x being cart's position) [1]. The authors suggested to select two "attractors" at $x = x_- = -\Delta/2$ and $x = x_+ = \Delta/2$, and to switch between these two attractors in a such a way that the pendulum energy increases. The control parameter Δ is chosen to decrease to zero as the pendulum approaches its inverted position. Since the acceleration of the cart is most effective when the pendulum crosses it pendent position, the authors used the following strategy. Starting near the attractor x_-, wait for the pendulum to cross lower vertical from right to left, then give short and violent acceleration and decelerate while the cart approaches other attractor x_+. The complete strategy consists in realizing a succession of such jerky transfers between the two attractors. The authors successfully implemented the method on a physical cart-pendulum apparatus but the main drawback of the method (and drawback of the Wei et al. method) is that it doesn't provide smooth acceleration/deceleration for the cart and thus places too much demand on the actuator.

1.4.4 Linear State-Space Control

The state-space control theory was ushered in at the USSR's successful launching of the first sputnik. The method become available to the control engineers after Luenberger developed the linear state observer in 1966 [33]. Since then the linear state-space control has proven to be the most robust method in application to control of unstable systems.

The first full state feedback controller for an inverted pendulum on a cart system, was implemented on a digital computer by Eastwood in 1968 [10]. This controller used linear state-space theory and elements of optimization in its design. Although direct

differentiation was used for velocity estimation, the system had satisfactory robustness.

In their inverted pendulum controller (1970) Bryson and Luenrberger combined the directly measured states with the velocity states estimated by linear state observer to effect control regulation.

By performing simulations of the nonlinear plant with linear control law Watts (1984) investigated the region of stability for the inverted pendulum model [63]. Henders and Soudack (1992) further investigated the region of stability of the nonlinear single inverted pendulum under of linear controller and predicted stable limit cycles in the cart's position [19].

Given below are a few general statements regarding linear state-space control for unstable systems.

- Satisfactory behavior of a linear state feedback controller can be achieved if an accurate mathematical model of the unstable system to be controlled is available. Linearizing this model about the operation point is the first step in the controller realization
- Full state information has to be available. The states can be either measured directly (with suitable measurement equipment) or recovered by state estimators. The implementation of state estimation approach also relies on an accurate linear model of the system's dynamics.
- The linear state space controller has proven to be the most robust method for

stabilizing an inverted pendulum system.

1.5 Thesis Layout

The thesis treats separately the pendulum swing-up and stabilization controllers. Three control approaches to swing up the pendulum are discussed: nonlinear state-space control, fuzzy control and an embedded artificial constraints control. The swing-up controllers are presented in the stated order as it is desired to provide a graduate transition to the method implemented physically on the cart-pole. For system stabilization with the pendulum in its inverted position linear state-space control is considered.

Chapter 2 describes the physical pendulum apparatus.

Chapter 3 begins with an introduction to nonlinear control theory. In order for a smoother transition to the more advanced nonlinear techniques to be established, the well-defined full state feedback linearization method is first presented. Discussed are the application constraints for that method in the case of pendulum swinging-up. With this foundation of nonlinear control, a nonlinear controller based on combination of the Lie theoretic approach and nonlinear output injection method is then defined and simulated.

Chapter 4 is dedicated to fuzzy logic control for pendulum swing up. The chapter first introduces the foundations of fuzzy logic and the key concepts involved in the controller design. The controller itself is designed as a Sugeno type fuzzy controller and is based on a compact inference system. The final form of the membership functions for the linguistic

variables for the selected system states is given. The presented controller simulation is followed by a discussion of its physical application.

Chapter 5 discusses the so-called Embedded Artificial Constraint Controller (EAC). The design of EAC treats pendulum energy issues and derives a suitable cart trajectory which provides the appropriate cart acceleration for gradually increasing the pendulum energy. The second part of the chapter presents the time domain design (root-locus design) of the cart position controller and discusses both its time and frequency domain behavior.

Chapter 6 discusses the inverted pendulum stabilizing controller. The chapter introduces the task of the linear state-space controller design. Velocity estimation theory is compared with evidence of its usefulness to high-speed digital implementation. Optimal controller realization is presented with discussion of its general application to unstable systems.

Chapter 7 addresses the software implementation of the controller. The chapter begins with description of the basic features of QNX Neutrino Real–Time Operating System and proceeds with the details of the realization of the client/server controller application.

Chapter 8 provides some concluding remarks on the control approaches presented.

Chapter 2

Pendulum Apparatus Specification Overview

The existing cart-pole system at the Advanced Control and Instrumentation Laboratory ay the University of Victoria provides an excellent platform for investigating control strategies on a physical under-actuated system.

The block diagram of the pendulum apparatus is given in Figure 2.1. The system consist of linear amplifier, DC motor and cart on a rail with pendulum feedback sensors. The controller is implemented on a digital computer running the QNX Neutrino 6.1 Real-Time Operating System. The controller computer is interfaced to the pendulum system through an optical encoder counter board and a digital to analog converter.

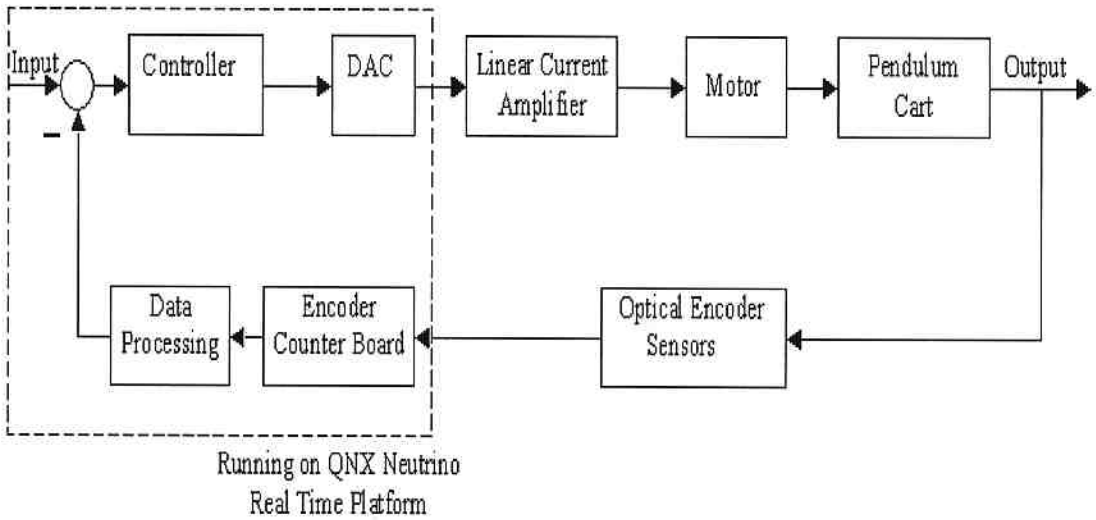


Figure 2.1 Pendulum Apparatus Block Diagram

2.1 Pendulum on a Cart System Equipment

The DC motor and Linear Amplifier

The motor and the linear amplifier are Electro-Craft Servo Products and are a matching pair in the sense that the manufacturer recommends this particular amplifier for the chosen permanent magnet DC motor.

The Linear Amplifier (model LA-5600) is capable of a peak output current of 22 Amps at 25 VDC. The bandwidth of the amplifier is 2 KHz and it is configured for torque control. The amplifier is controlled by a + / - 10 VDC input control signal. The LA-5600 is equipped with AC input line transformer, an integral DC power supply and provides circuitry which protects the motor from overloading. In its operating range (when not saturated) the linear amplifier acts as linear voltage controlled current amplifier with a gain of 2.2 A/V.

The motor (Electro-Craft model #0586-01-047) is a high performance permanent magnet DC motor. It has a no-load rating for up to 6000 rpm. The published parameters of the motor are given in Appendix C, table C.1.

Optical Encoders

The cart position and pendulum angle are measured directly by two optical encoders mounted at motor shaft and pendulum joint. The encoders have 4096 and 2048 counts per revolution, respectively. Considering setup and gearing this gives a resolution of 39 μm for thr cart position and 3.06×10^{-3} rad (0.18°) for the pendulum angle.

The resolution of both of the encoders assures satisfactory performance for the swing-up and stabilization controllers.

2.2 PC Based Instrumentation

Optical Encoder Counter Board

The quadrature encoded signals from the optical encoders are decoded by the optical board. The board is a KEITHLEY model M5312-4 operating over the PC's ISA bus. The board provides inputs for four encoders and can perform quadrature decoding at speeds up to 333 KHz. Inputs are conditioned by a four-stage digital filter which assures that the signals from the encoders are valid for four cycles of the software adjustable sampling clock. The numerical values corresponding to the current encoder's position can be obtained by the software through the respective board registers.

DAC

The criteria for selecting the *Digital to Analog Converter* (DAC) board were good analog output signal resolution and a sampling rate of at least 1 KHz. The chosen board is a National Instruments ISA board – model AT-MIO-16E-10. The Digital to Analog Converter provides 16 bit analog conversion which in the case of +/-10V bipolar output corresponds to resolution of 0.31 mV at a sampling rate at up to 100 kHz. This is enough to assure satisfactory behavior for both the swing-up and the stabilization controller.

Physical Parameters

The length of the cart rail is 0.8 m. The effective mass M of the cart is 0.8093 kg and the viscous friction coefficient for horizontal motion of the cart b_c is 2.4 N·s/m. The total mass of the pendulum rod and the cylinder at the end of the rod is 0.214 kg. The moment of inertia of the pendulum rod and cylinder about their center of mass is calculated to be

$Jp = 0.043$ kg·m^2. The viscous friction coefficient for the angular motion of the pendulum is $b_p = 0.001$ N·s.

Complete list of the system's physical parameters is given in Appendix.

Pictured in Figure 2.2 is the complete pendulum on a cart apparatus.

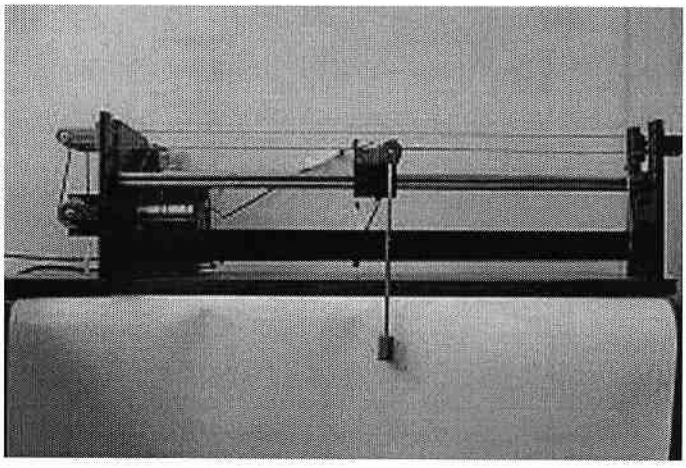


Figure 2.2 Pendulum on a Cart Apparatus

Chapter 3

Nonlinear State-Space Controller

3.1 Chapter Introduction

Several efforts have been made for implementing a nonlinear pendulum swing-up controller. Most of these works employ some approximation to the nonlinear terms, [27], otherwise they use controllers based on the trajectory for swing-up beforehand by numerical simulations.

The purpose of this chapter is to illustrate the feasibility of the derivation of a nonlinear control method for swinging up a pendulum, which considers the entire system dynamics without using any approximations to the nonlinear terms.

Strictly referring to the nonlinear control theory developed by [21], [25], [56] the pure control of non-linear systems should be done by the so-called feedback linearization. The basic idea of this approach is to construct a non-linear control law as a so called inner loop control which, in the ideal case, exactly linearizes the nonlinear system after a suitable state space change of the coordinates. The designer can then design a second stage or an outer loop control in the new coordinates to place the poles of the system in the desired locations.

This chapter first presents the foundation of the nonlinear theory giving the necessary definitions and formulations, and then proceeds to describe nonlinear control methods.

Considered first is the exact feedback linearization method. The theory behind this method is well developed [21, 25] and with a straightforward application, the description of that approach is a perfect way of introducing the concept of the feedback linearization. The method's applicability issues for the pendulum swing-up problem are discussed.

Presented after that is a feedback linearization method combining several nonlinear control strategies. The method is directly applicable to the cart-pole system and realizes both the swinging-up and the stabilization of the pendulum. Simulations and comments on the method are presented.

3.2 System Dynamics and State-Space Representation

The dynamic equations of motion of the cart-pendulum system as derived in Appendix B are:

$$(M+m)\ddot{x}_M + ml_g\ddot{\theta}\cos\theta - mgl_g\dot{\theta}^2\sin\theta + b_c\dot{x}_M = \frac{A_G T_t}{r}u \tag{3.1}$$

$$\left(ml_g^2 + J_p\right)\ddot{\theta} + ml_g\ddot{x}_M\cos\theta - mgl_g\sin\theta + b_p\dot{\theta} = 0 \tag{3.2}$$

with x_M and θ being the cart and pendulum position (Figure 3.1).

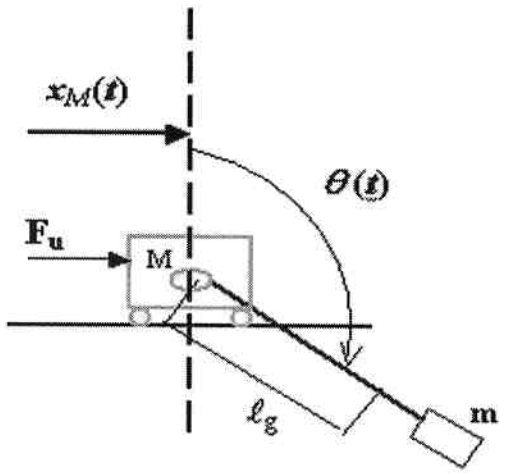


Figure 3.1 Pendulum on Cart system for Nonlinear Control Modeling

By solving (3.1) and (3.2) with respect to $\ddot{x}_M$ and $\ddot{\theta}$:

$$\ddot{x}_M = \frac{J_0(ml_g\dot{\theta}^2 \sin\theta - b_c\dot{x}_M) - ml_g \cos\theta(mgl_g \sin\theta - b_p\dot{\theta})}{M_0 J_0 - m^2 l^2 \cos^2\theta} +$$

$$+\frac{A_g T_t / rJ_0}{M_0 J_0 - m^2 l^2 \cos^2\theta} \equiv f_3(\theta, \dot{x}_M, \dot{\theta}) + g_3(\theta)u \tag{3.3}$$

$$\ddot{\theta} = \frac{1}{J_0}\left[-ml_g \cos\theta f_3(\theta, \dot{x}_M, \dot{\theta}) + mgl_g \sin\theta - b_p\dot{\theta}\right] \equiv$$

$$\equiv f_4(\theta, \dot{x}_M, \dot{\theta}) + g_4(\theta)u, \tag{3.4}$$

where $M_0 = M + m$ and $J_0 = J_p + ml_g^2$

To derive the state-space representation of the system, the state vector **x**(t) is defined as:

$$\mathbf{x}(t) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_M(t) \\ \theta(t) \\ \dot{x}(t) \\ \dot{\theta}(t) \end{bmatrix} \tag{3.5}$$

then

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ f_3(x_2, x_3, x_4) \\ f_4(x_2, x_3, x_4) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g_3(x_2) \\ g_4(x_2) \end{bmatrix} u \tag{3.6}$$

or, in vector form

$$\dot{\mathbf{x}} = \mathbf{f(x)} + \mathbf{g(x)}\, u \tag{3.7}$$

where the definitions of the two vector-valued functions **f(x)** and **g(x)** are clear.

The cart position and pendulum angle define the output of the system:

$$\mathbf{y} = \begin{bmatrix} x_M(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.8}$$

3.3 Definitions Used in The Nonlinear Control Theory

This section provides the necessary definitions for further development.

Definition 3.1 *Lie derivative* [21]

Let h: $\mathbf{IR}^n$ -> **IR** be a scalar function. The gradient of h denoted by **dh**, is the row vector

$$\mathbf{dh} = \left[\frac{\partial h}{\partial x_1}, \ldots, \frac{\partial h}{\partial x_n} \right] \tag{3.9}$$

For a scalar function h, the **Lie derivative** along the vector field $\mathbf{f} = (f_1, \ldots, f_n)$ is defined as:

$$L_{\mathbf{f}} h(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} f_i(\mathbf{x}) \tag{3.10}$$

Inductively

$$L_{\mathbf{f}}^{k} h(x) = L_{\mathbf{f}}\left(L_{\mathbf{f}}^{k-1} h(x)\right) \text{ with } L_{\mathbf{f}}^{0} h(x) = h(x) \tag{3.11}$$

Definition 3.2 *Lie Bracket* [56]

Let **f** and **g** be two vector fields on $\mathbf{IR}^n$. The **Lie Bracket** of **f** and **g**, denoted by [**f**, **g**] is a third vector field defined by:

$$[\mathbf{f},\mathbf{g}] = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\mathbf{f} - \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{g} \tag{3.12}$$

where $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ *(respectively* $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$*) denotes the n x n Jacobian matrix whose ij – th entry is* $\frac{\partial g_i}{\partial x_j}$ *(respectively* $\frac{\partial f_i}{\partial x_j}$*).*

The $[\mathbf{f}, \mathbf{g}]$ is denoted as $\mathbf{ad_f(g)}$ and $\mathbf{ad}_{\mathbf{f}}^{k}(g)$ is defined inductively by

$$\mathbf{ad}_{\mathbf{f}}^{k}(\mathbf{g}) = [\mathbf{f}, \mathbf{ad}_{\mathbf{f}}^{k-1}(\mathbf{g})] \tag{3.13}$$

Definition 3.3 *Involutive Set* [56]

A linearly independent set of vector fields $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$is said to be *involutive* if and only if there exist scalar functions $\alpha_{ijk} : \Re^n \to \Re$ such that

$$[\mathbf{X}_i, \mathbf{X}_j] = \sum_{k=1}^{m} \alpha_{ijk}\mathbf{X}_k \text{ ,for all } i, j, k \tag{3.14}$$

Involutivity simply means that a Lie Bracket of any pair of vector fields from the set $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$ results in a vector field that can be expressed as a linear combination of the original vector fields $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$.

Definition 3.4 *Diffeomorphism* [21]

A nonlinear change of coordinates

$$\mathbf{z} = \mathbf{T}(\mathbf{x}) \tag{3.15}$$

Where $\mathbf{x} = [x_1, \ldots, x_n]$ and $\mathbf{T}(\mathbf{x}) = [T_1(\mathbf{x}) \;\; T_2(\mathbf{x}) \;\; \ldots \;\; T_n(\mathbf{x})]^T$

with the following properties

(i) $\mathbf{T}(\mathbf{x})$ is invertible, i.e. there exists the function $\mathbf{T}^{-1}(\mathbf{z})$ such that

$$\mathbf{T}^{-1}(\mathbf{T}(\mathbf{x})) = \mathbf{x}$$

for all $\mathbf{x}$ in $\mathbf{IR}^n$.

(ii) $\mathbf{T}(\mathbf{x})$ and $\mathbf{T}^{-1}(\mathbf{z})$ are both smooth mappings, i.e. they have continuous partial derivatives of any order.

A transformation of this type defines *diffeomorphism* on $\mathbf{IR}^n$. The first of the two properties is needed in order to provide the possibility of reversing the transformation and recovering the original state vector as:

$$\mathbf{x} = \mathbf{T}^{-1}(\mathbf{z})$$

while the second one guarantees that the description of the system in the new coordinates is still a smooth one.

3.4 Exact Feedback Linearization Method

This subsection introduces the method of exact feedback linearization and discusses its feasibility for the pendulum swing –up problem.

A single input system of order n of the form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u \tag{3.16}$$

(which is the case of inverted pendulum) where **f**(x) and **g**(x) are smooth vector fields is *exact feedback linearizable* if there exists a region U in $\mathbf{IR}^n$ containing the origin, a diffeomorphism T U-> $\mathbf{IR}^n$, and a nonlinear feedback

$$u = \alpha(\mathbf{x}) + \beta(\mathbf{x})v \tag{3.17}$$

with $\beta(\mathbf{x}) \neq 0$ on U such that the transformed variables

$$\mathbf{z} = \mathbf{T}(\mathbf{x}) \tag{3.18}$$

satisfy the system equation

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{b}v \tag{3.19}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & \dots & & \\ 0 & 0 & \dots & & 1 \\ 0 & 0 & \dots & & 0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ \\ 1 \end{bmatrix} \tag{3.19}$$

As derived in [56] the diffeomorphism $\mathbf{T}(\mathbf{x}) = [T_1(\mathbf{x}) \dots \quad T_n(\mathbf{x})]^T$ can be found by first determining T_1 from the system of partial differential equations:

$$L_{ad_{\mathbf{f}}^{k}(\mathbf{g})} T_1(\mathbf{x}) = 0;\ k = 0,1,\ldots\ldots,n-2$$
$$L_{ad_{\mathbf{f}}^{n-1}(\mathbf{g})} T_1(\mathbf{x}) \neq 0 \tag{3.20}$$

and then $T_2(\mathbf{x}),\ldots., Tn(\mathbf{x})$ can be inductively found from:

$$T_{i+1}(\mathbf{x}) = L_{\mathbf{f}}\, T_i(\mathbf{x}); i = 1,\ldots,n-1 \tag{3.21}$$

and the control input u is found as:

$$u = \frac{1}{L_{\mathbf{g}} T_n(\mathbf{x})}(-L_{\mathbf{f}} T_n(\mathbf{x}) + v) \tag{3.22}$$

Thus the exact feedback linearization problem is reduced to solving the system (3.20) for $T_1(\mathbf{x})$. When does such a solution exist? The following theorem provides the necessary and sufficient conditions for existence of an exact feedback linearization.

Theorem 3. 1 *Exact feedback linearization existence*

The nonlinear system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u$$

with $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ being smooth vector fields, and $\mathbf{f}(0)=0$ is feedback linearizable if and only if there exists a region U containing the origin in $\mathbf{IR}^n$ in which the following conditions hold:

1. The rank of the matrix $\left\lfloor \mathbf{g}, \mathbf{ad_f}(\mathbf{g}), \ldots, \mathbf{ad_f}^{n-1}(\mathbf{g}) \right\rfloor$ is equal to the degree of the system n.
2. The set $\left\{ \mathbf{g}, \mathbf{ad_f}(\mathbf{g}), \ldots, \mathbf{ad_f}^{n-2}(\mathbf{g}) \right\}$ is involutive in U.

The proof of the theorem is given in [21]

The summarized procedure for the exact feedback linearization is

1. Present the system in a state space form.
2. From (3.20) and (3.21) find the diffeomorhism $\mathbf{T}(\mathbf{x})$ necessary for calculation of the new coordinates.
3. For the linear system (3.19) at each time t find the external control input v by some control technique (e.g. pole placement, *LQR* method).
4. From (3.22) calculate the nonlinear control input u.

The process is illustrated in Figure 3.2

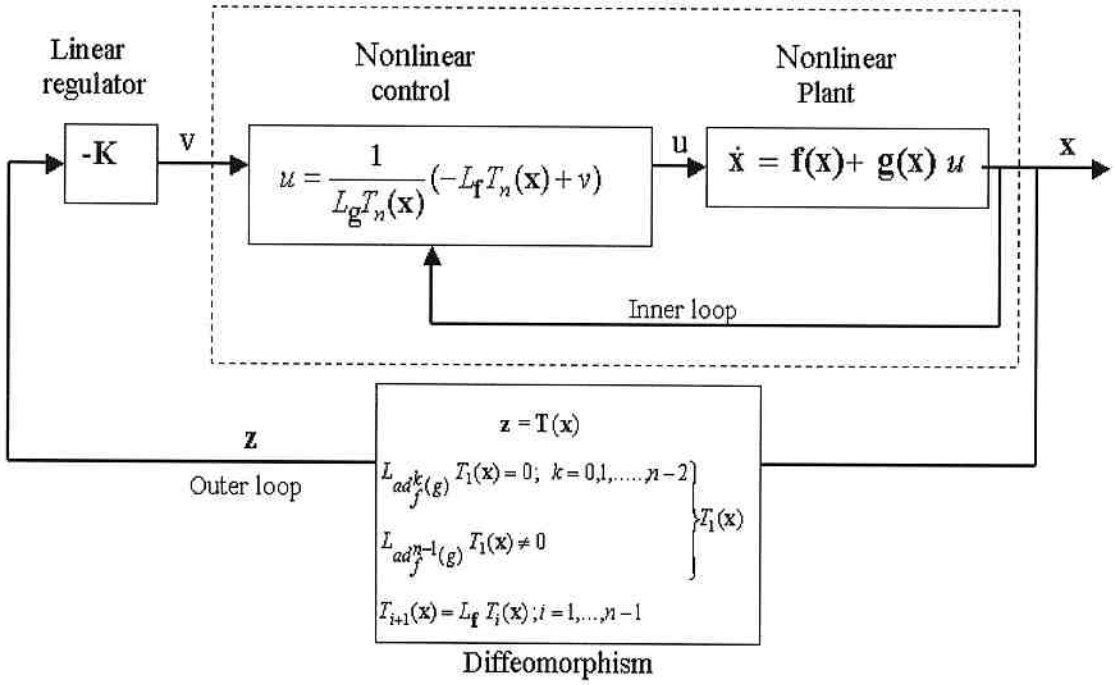


Figure 3.2 Exact Feedback Linearization Block Diagram

In the case of the pendulum on a cart system, it has been determined through simulations that the rank of the matrix $\left[\mathbf{g}, \mathbf{ad_f}(\mathbf{g}), \mathbf{ad_f^2}(\mathbf{g}), \mathbf{ad_f^3}(\mathbf{g})\right]$ is equal to the system's order of 4 and thus the first condition in Theorem 3.1 is satisfied but the distribution $\left\{\mathbf{g}, \mathbf{ad_f}(\mathbf{g}), \ldots, \mathbf{ad_f^{n-2}}(\mathbf{g})\right\}$ is not involutive in U [5]. That is, the necessary and sufficient condition is not fulfilled and exact feedback linearization doesn't exist. Or in other words – although the theory of the exact feedback linearization is well defined and with good results in applications, the method can not be used to swing up the pendulum from its initial (downwards) to its inverted position.

3.5 Nonlinear Control with Feedback Linearization and Nonlinear Output Injection

As it has been shown in the previous subsection the concept of exact feedback linearization can't be applied to swing up the inverted pendulum from its downwards position to its inverted position. Several different approaches presented in the literature have been investigated [5], [64], [54] in an attempt to find a suitable nonlinear controller for the cart-pendulum system to be formed. Most of them use more or less simplifying assumption for the system dynamics, which result in a given approximation of the nonlinear terms.

The approach that has been considered to be the most suitable for implementation on the physical pendulum cart system is the one taken in [42]. This approach considers the exact dynamics of the system without any approximations and at the same time is quite elegant. This method with some modifications is the basis of the following presentation and following simulations.

3.5.1 Relative System Degree

The analysis of nonlinear systems as approached by Isidori [21] relies on the concept of the relative degree of the system.

The concept of relative degree is determined in accordance with the *output* function of the system $h(\mathbf{x})$. The function $h(\mathbf{x})$ is a scalar function of the system states that form its output. With reference to (3.8):

$$h(\mathbf{x}) = h(x_1, x_2) \tag{3.23}$$

According to [21] the nonlinear system (3.7) is said to have relative degree r at a point in the state space $\mathbf{x}^0$ if

$$\begin{aligned} &L_{\mathbf{g}} L_{\mathbf{f}}^{n} h(\mathbf{x}) = 0, \;\; k = 0,1,2,\ldots,r-2 \\ &L_{\mathbf{g}} L_{\mathbf{f}}^{r-1} h(\mathbf{x}) \neq 0 \end{aligned} \tag{3.24}$$

for all $\mathbf{x}$ in the neighborhood of $\mathbf{x}^0$

By using definition of Lie Derivative (Definition 1), for $n = 0$ follows:

$$L_{\mathbf{g}} L_{\mathbf{f}}^{0} h(\mathbf{x}) = L_{\mathbf{g}} h(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} f_i(\mathbf{x}) = \frac{\partial h(\mathbf{x})}{\partial x_3} g_3(x_2) + \frac{\partial h(\mathbf{x})}{\partial x_4} g_4(x_2) \tag{3.25}$$

Now, since $h(\mathbf{x})$ is independent of x_3 and x_4, $L_g L_f^0 h(x)$ becomes identically zero no matter what the nonlinear functions $g_3(x_2)$ and $g_4(x_2)$ are and no matter how $\mathbf{x}^0$ is selected.

For n=1 :

$$L_{\mathbf{g}} L_{\mathbf{f}} h(\mathbf{x}) = L_{\mathbf{g}}\left(L_{\mathbf{f}} h(\mathbf{x})\right) = L_{\mathbf{g}}\left(\frac{\partial h(\mathbf{x})}{\partial x_1} x_3 + \frac{\partial h(\mathbf{x})}{\partial x_2} x_4\right) = \\ = \frac{\partial h(\mathbf{x})}{\partial x_1} g_3(x_2) + \frac{\partial h(\mathbf{x})}{\partial x_2} g_4(x_2) \tag{3.26}$$

which is not equal to zero since $h(\mathbf{x}) = h(x_1, x_2)$ and thus the relative degree of the pendulum-cart system (3.7) is n=2.

According to [21] for a system with a relative degree equaling two, the state feedback control that results in a stable closed-loop partly linearized system must be chosen as [21]

$$u = \frac{1}{L_{\mathbf{g}} L_{\mathbf{f}} h(\mathbf{x})}(-L_{\mathbf{f}}^2 h(\mathbf{x}) + v) \tag{3.27}$$

where $v(t)$ is an external input to be designed as a control signal.

For constructive simplicity, the function $h(\mathbf{x})$ in the further derivation, is chosen to be $h(\mathbf{x}) = x_1$. Equally $h(\mathbf{x})$ can be selected as the other output coordinate, $h(\mathbf{x}) = x_2$, but this results in a more complicated system although the analysis is the same [42]. If the output function is selected as a linear combination of the output coordinates (e.g. $h(\mathbf{x}) = x_1 + x_2$) then the nonlinear theory as developed by [21] suggests that the new coordinates be presented as $h(x_1, x_2)$, $L_f h(x_1, x_2)$ and that the resulting system be analyzed in a different approach that the one presented below. Thus this case will not be considered.

It should be noted that by choosing $h(\mathbf{x}) = x_1$ the nonlinear cart-pendulum system given by (3.7) has a relative degree of two no matter how the point in the state space $\mathbf{x}$ is selected and thus the system has a *global* relative degree of two and the control signal (3.27) is valid globally.

With $h(\mathbf{x}) = x_1$, (3.27) yields:

$$u = -\frac{f_3(x_2, x_3, x_4)}{g_3(x_2)} - \frac{1}{g_3(x_2)} v, \tag{3.28}$$

Now, with (3.28) the system dynamics (3.6) becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ 0 \\ -\frac{1}{J_0}(b_p x_4 - mgl_g \sin x_2) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ -\frac{1}{J_0} ml_g \cos x_2 \end{bmatrix} v \tag{3.29}$$

and the nonlinear feedback control u is

$$u = \frac{1}{A_g T_t / r}\left\{\frac{1}{J_0} m l_g \cos x_2 \left(m g l_g \sin x_2 - b_p x_4\right) - \right.$$

$$\left. - m l_g x_4^2 \sin x_2 + b_c x_3 + \left(M_0 - \frac{1}{J_0} m^2 l_g^2 \cos^2 x_2\right) v \right\} \qquad (3.30)$$

3.5.2 Linearizations

In the previous subsection, the form of the feedback control for the pendulum swing-up and stabilization problem was derived as (3.30). The signal v(t), which appears in (3.30), is designed as a feedback control for the linear system model obtained from the nonlinear system (3.7). For the purpose of linearization the approach of *linearization by nonlinear output injection* as presented in [30] is followed .

Let us define

$$v_0 = v - x_2 \qquad (3.31)$$

Comments on this coordinate change are given below.

With (3.31), (3.29) can be expressed as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -bp/J_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \\ + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -ml_g(x_2 \cos x_2 - g \sin x_2)/J_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ -ml_g \cos x_2 / J_0 \end{bmatrix} v_0 \tag{3.32}$$

Let us assign q=x_2. Then (3.32) can be expressed in vector form as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{p}(q) + \mathbf{r}(q)v_0 \tag{3.33}$$

This equation has the form of nonlinear output injection to a linear system, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ [30]. By introducing a scalar signal $\tilde{v}(t)$ and a constant vector **b** such that

$$\mathbf{p}(q) + \mathbf{r}(q)v_0 = \mathbf{b}\tilde{v} \tag{3.34}$$

for each time t, (3.33) becomes the linear system:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}\tilde{v} \tag{3.35}$$

The vector **b** should be selected in a way to guarantee the controllability of (3.35). One such form of b is $\mathbf{b}=[0\ 0\ 1\ 1]^T$. The transformation (3.31) was introduced to ensure existence of a controllable form of (3.35). Without (3.31) controllability of (3.35) is no longer guaranteed no matter how **b** is selected.

The signal $\tilde{v}(t)$ should swing up the pendulum to its inverted position and then balance it (i.e. maintain the state vector $\mathbf{x}\approx 0$). Thus, it can be designed as a state-space feedback control signal for the system (3.35):

$$\tilde{v} = -\mathbf{G}\mathbf{x} \tag{3.36}$$

The matrix **G** can be determined by direct pole-placement algorithms, or by applying the Linear Quadratic Regulator (LQR) method (a detailed *pole-placement* discussion including description of the LQR is given in Chapter 6)

In summary, the procedure for swinging-up the pendulum by partial feedback linearization method involves the following steps at each time *t:*

1. Calculate the control signal $\tilde{v}(t)$ from (3.36)
2. Calculate $v_0(t)$ from (3.34) as:

$$v_0(t) = \frac{1}{\mathbf{r}^T(q)\mathbf{r}^T(q)}\mathbf{r}^T(q)\left[b\tilde{v} - \mathbf{p}(q)\right] \tag{3.37}$$

3. Calculate $v(t)$ from (3.31)
4. Calculate the nonlinear control signal u(t) from (3.30)

The block diagram in Figure 3.3 depicts the overall form of the controlled cart-pole system:

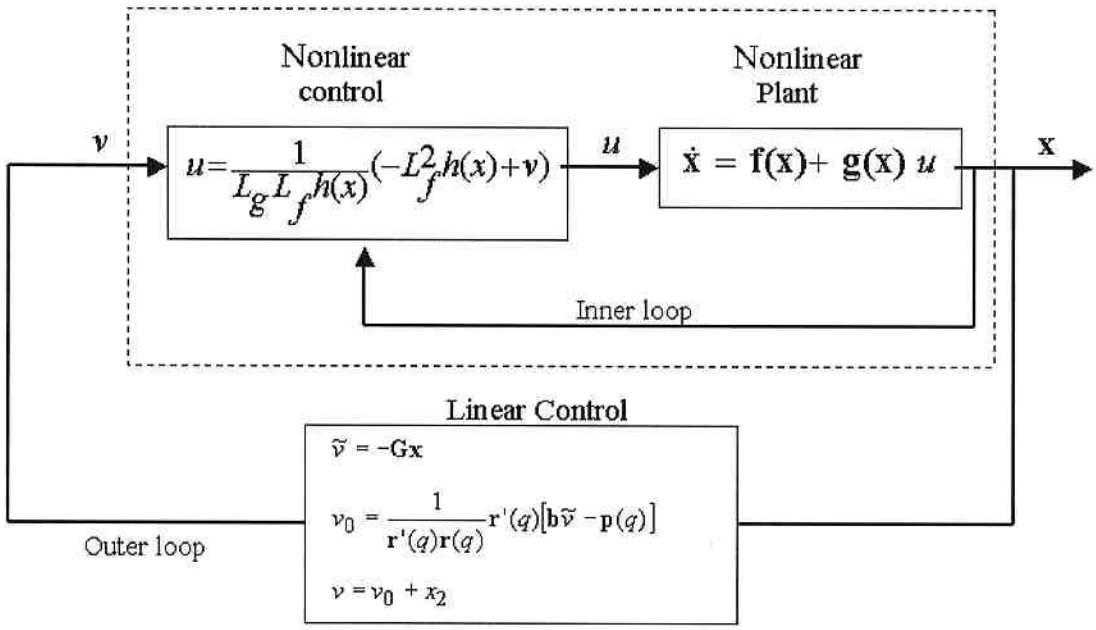


Figure 3.3 Proposed Nonlinear Controller Block Diagram

3.5.3 Controller Implementation Considerations

The goal of the control action (3.36) is to swing up and stabilize the pendulum in the position $\theta = 2k\pi$. With reference to Figure 1, if the pendulum is swinging upwards (or rotates counterclockwise) and thus $\dot{\theta}(t) < 0$ (or $\dot{\theta}(t) > 0$ when $\theta(t) < 0$) the most

effective way to swing the pendulum up is to apply control such that the angle $\theta(t)$ becomes zero.

The case when the pendulum is swinging downwards (i.e. $\theta(t) > 0 \text{ and } \dot{\theta}(t) > 0 \text{ or } \theta(t) < 0 \text{ and } \dot{\theta}(t) < 0$) can be divided into two sub-cases. First, if the pendulum is not yet swinging down beyond a selected threshold angle θtr, that is $|\theta(t)| <= \theta_{tr}$, then the destination angle is $\theta=0$.

Once the threshold is passed it is more reasonable the pendulum to be controlled in a way such that $\theta(t)$ becomes 2π rad (or -2π if $\theta(t)<0$). It should be noted that the control signal (3.36) is designed to keep $\theta\approx 0$. To overcome this contradiction $\theta(t)$ can be equally replaced by $-2\pi+\theta(t)$ ($2\pi+\theta(t)$, when $\theta(t) <0$) making the destination angle 0. This will lead to replacing x_2 ($=\theta$) in (3.32) by $\pm 2\pi+x_2$.

Considering the direction of swinging of the pendulum and whether or not the threshold angle is passed, the modifications in (23) are as follows [42]

(i) If $\theta(t) > 0 \text{ and } \dot{\theta}(t) < 0 \text{ or } \theta(t) < 0 \text{ and } \dot{\theta}(t) > 0$ no modifications are necessary

(ii) If $\theta(t) > 0 \text{ and } \dot{\theta}(t) > 0$ - the substitution $-2\pi+x_2$ for x_2, affects the fourth equation in (23) is changed, so that

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \{\mathbf{p}(q) + \mathbf{p}_m(q)\} + \mathbf{r}(q)v_0 \tag{3.38}$$

where

$$\mathbf{p}_m(q) = \begin{bmatrix} 0 \\ 0 \\ -\dfrac{2}{\pi} \\ \dfrac{2\pi}{J_0} ml \cos x_2 \end{bmatrix} \tag{3.39}$$

(iii) If $\theta(t) < 0$ and $\dot{\theta}(t) < 0$ - similar to case (ii) the substitution $-2\pi+x_2$ for x_2, yields:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \{\mathbf{p}(q) - \mathbf{p}_m(q)\} + \mathbf{r}(q)v_0 \tag{3.40}$$

If $p_0(y)$ is defined as:

$$\mathbf{p}_0(y) = \begin{cases} \mathbf{p}(q) & \text{for } case\,(i) \\ \mathbf{p}(q) + \mathbf{p}_m(q) & \text{for } case\,(ii) \\ \mathbf{p}(q) - \mathbf{p}_m(q) & \text{for } case\,(iii) \end{cases} \tag{3.41}$$

then the modifications are performed by replacing $\mathbf{p}(q)$ in (3.34) by $\mathbf{p}_0(q)$ and the control signal (3.36) can be designed by applying the LQR method for the linear system (3.34).

3.5.4 Some Comments on the Observer

The realization of the nonlinear controller as described above is done with the assumption that the whole state vector is available. Physically, only the pendulum angle θ and cart

position x_M are measured. The remaining two states must be recovered. Design of a non-linear observer is beyond the scope of this thesis.

A possible solution could be to derive a linear observer for the pendulum–cart system which is linearized about the inverted position of the pendulum ($\mathbf{x}$=0) and to examine the observer's system matrix at every angular displacement of the pendulum [24].

The linear observer is defined as [16]

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}u + \mathbf{L}(\mathbf{y} - C\hat{\mathbf{x}}) \tag{3.42}$$

where $\hat{\mathbf{x}}$ is the observed state vector

The output vector is

$$\mathbf{y} = \mathbf{C}x = \begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.43}$$

and thus

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.44}$$

The matrices **A** and **B** are obtained by linearizing the system model (3.6) about $\mathbf{x}$=0:

Noting that (3.6) can be presented as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u = \mathbf{F}(\mathbf{x}, u) \tag{3.45}$$

then

$$\mathbf{A} = \frac{\partial \mathbf{F}(\mathbf{x}, u)}{\partial \mathbf{x}}\Big|_{\mathbf{x}=0, u=0}$$

and (3.46)

$$\mathbf{B} = \frac{\partial \mathbf{F}(\mathbf{x}, u)}{\partial u}\Big|_{\mathbf{x}=0, u=0}$$

Then the observer's gain matrix is determined such that the eigenvealues of the observer system matrix [**A-LC**] have the desired values. This method is applied to the nonlinear model observer. It can be shown [24] that with the already determined **L**, [**A-LC**] has eigenvalues with negative real parts when **A** is linearized everywhere in the state-space and thus the observer is a stable system. If the observer designed this way shows an unsatisfactory behavior and a suitable nonlinear controller is unavailable, then a simple differentiation in time can be attempted for recovering the cart and pendulum velocities during the swinging-up of the pendulum. Once the pendulum starts to exhibit small deviations from its inverted position a linear model observer should be utilized (linear observers are presented in details in Chapter 6).

3.5.5 Simulations

Continuous simulations have been implemented using the Runge-Kutta method in Matlab's Ordinary Differential Equation Solver. The script file is listed in Appendix D - *nonlinear_controller.m*

For the design of the Linear Quadratic Regulator for the linear system (3.35) the weight matrices **Q** and **R** have been defined as:

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \qquad \mathbf{R} = [5] \tag{3.47}$$

which produces the controller gain matrix **K** as:

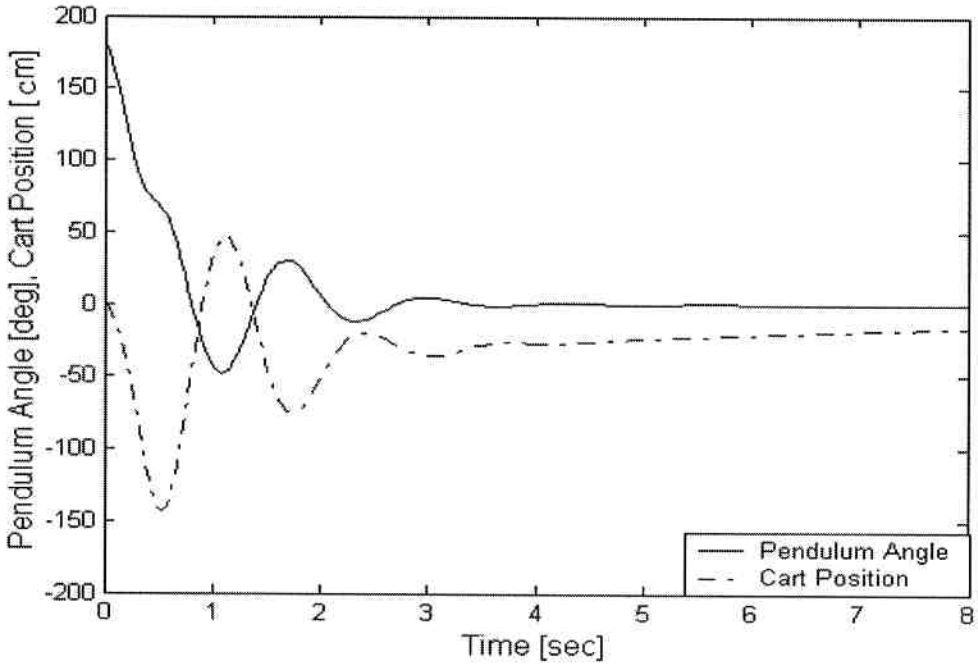
$$\mathbf{K} = [-0.45 \;\; -15.05 \;\; -3.69 \;\; -3.90] \tag{3.48}$$

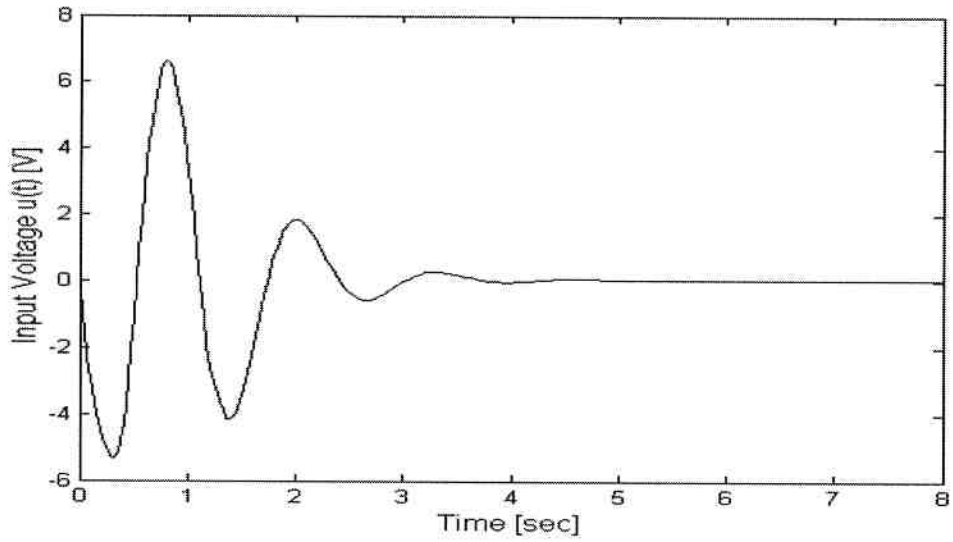
And thus the closed loop poles of the linear system (3.35) with the input (3.36) are:

$$poles = [-5.0456 \pm j5.0228 \quad -0.1273 \pm j0.1242\mathrm{i}] \tag{3.49}$$

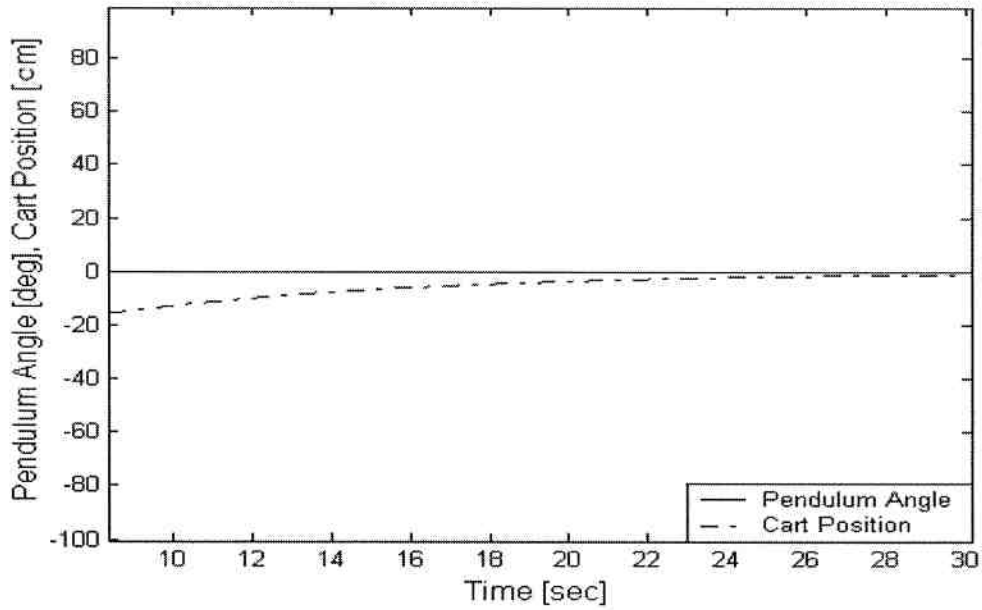
For the simulation purposes the threshold angle θ_{tr} as defined in section 3.5.3 was selected as $\theta_{tr} = 45°$

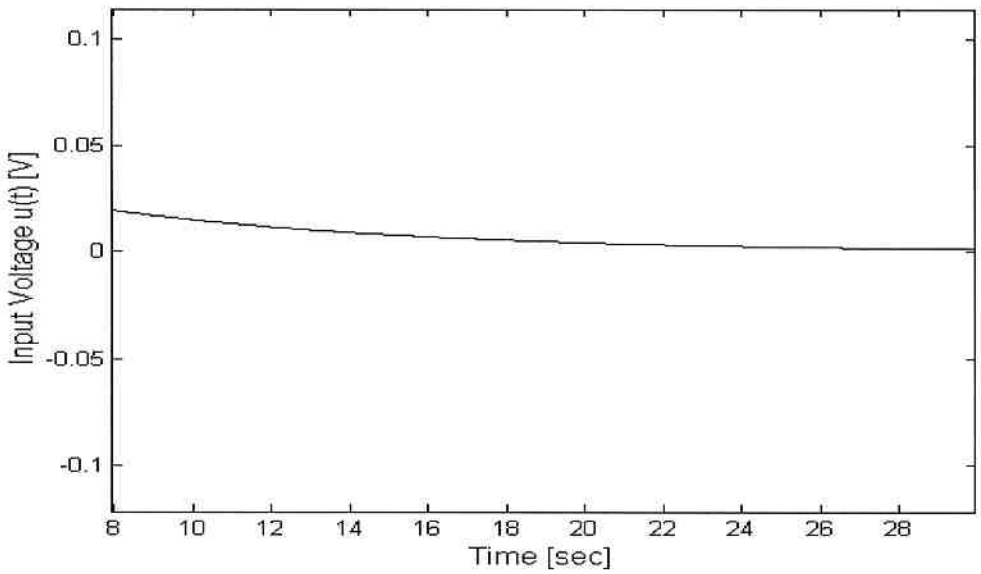
The simulations results are presented in the two graph pairs (Graph 3.1 and Graph 3.1b) with the second pair showing the convergence of the system output and the control input.


Pendulum Angle [deg], Cart Position [cm]
Time [sec]
Pendulum Angle
Cart Position


Input Voltage u(t) [V]
Time [sec]

Graph 3.1a





Graph 3.1b

Graph 3.1 Proposed Nonlinear Controller Simulations

3.5.6 Comments on the Nonlinear State-Space Controller

The simulations given above illustrate the feasibility of the suggested nonlinear controller. The control signal is in the acceptable –10 to +10V range. The constraint that prevents the implementation of the controller on the existing pendulum apparatus at the University of Victoria control laboratory is the limited to 0.8m cart rail length. And as seen from figure there is a requirement that the cart travel approximately between –1.5m and 0.5m (or a deviation of 2 meters) until the pendulum is stabilized. The controller behavior depends on the selection of the locations of the closed-loop poles of the linear system (respectively the values of the elements of the controller gain matrix **G**). Thus, with the selected **G** matrix the pendulum reaches its inverted position after a single swing. Different pole locations (and thus different forms of **G**) have been attempted which lead to swinging the pendulum more than once but still the cart excursion remained more than 1.4 m. The best controller performance during the simulations (in terms of system behavior and demand on the actuator) is the one presented above.

3.6 Chapter Conclusions

Control of under-actuated nonlinear systems is, in general, a rather challenging task. Several nonlinear control methods have been investigated for their applicability to the pendulum swing-up problem. Being a system possessing fewer actuators than its degree of freedom, the cart pendulum system cannot be controlled by the exact feedback linearization control method.

The controller approach which has been thoroughly studied and considered for possible implementation, combines several nonlinear control techniques described in the literature. A complete simulation model for the controller has been built. The system showed satisfactory behavior under the proposed controller. The physical limitation of the rail length of the existing pendulum apparatus has prevented conducting actual experiments.

Chapter 4

Fuzzy Swing-Up Controller

4.1 Chapter Introduction

The conventional approach for synthesis of the control strategy and the controller itself is based on the derivation of a mathematical model of the object or process under control. The models of an object under control involve quantitative, numerical calculation and are commonly constructed in advance, before realization. In adaptive controllers, the exact control strategy is not calculated in advance but is generated by optimization algorithms based on the controller experience. However this approach also assumes the construction of numeric mathematical models.

Unlike conventional state-space and adaptive controllers, a fuzzy controller can be linguistically conceived to provide swinging up of the pendulum from its initial position without the knowledge of the underlying equations of motion. The strength of the fuzzy – logic controller design lies in the ability to conceive a controller without having to model the system in terms of its governing differential equations. The cost of modeling the system in state-space control methods is traded for the task of linguistically determining a set of control rules in the fuzzy control approach.

4.2 Fuzzy Logic Foundations

This section provides the theoretical background in the fuzzy logic field in terms of definitions and models, necessary for the implementation of the fuzzy controllers.

The purpose of fuzzy logic is to map an input space to an output space, and the primary mechanism for doing this is a list of *if-then* statements called rules. All rules are evaluated in parallel, and the order of the rules is unimportant. The rules themselves are useful because they refer to variables and the adjectives that describe those variables. Before building the system that interprets rules, all input and output variables have to be defined in terms of their range and the adjectives that describe them – that is a linguistic model of the system is needed

Definition 4.1 *Fuzzy Set* [40]: A fuzzy set A in the universal set U is a set of ordered pairs of a generic element u and its membership degree $\mu_A(u) \in [0, 1]$ as $A=\{ (u, \mu_A(u)) / u \in U \}$.

As it can be seen from Definition 4.1, the fuzzy set is given by its membership function. The value of this function (as discussed above) determines if the element belongs to the fuzzy set and in to what degree.

The types and properties of the membership functions are discussed in the next section.

4.2.1 Membership Functions

A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the universe of discourse.

The most popular forms of membership functions are the *triangular*, the *trapezoidal*, the Gaussian, and, the *generalized bell* functions. The first two consists just of straight lines. The third and fourth are built respectively over the Gaussian distribution curve: $\mu(x)=exp(-(x-\lambda)^2 / 2\sigma^2)$ and the generalized bell function - $\mu(x) = 1/|(x-c)/a|^{2b}$.

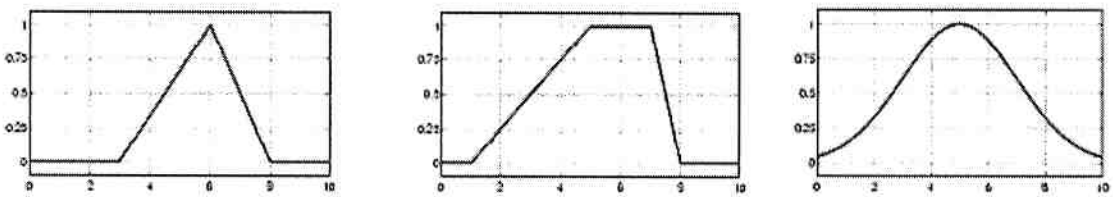


Figure. 4.1 Common Membership Functions

The straight-line membership functions have the advantage of simplicity and require less computational power. On the other hand membership functions based on the Gaussian distribution and generalized bell curves provide smoothness.

4.2.2 Linguistic variables

A linguistic variable is defined by < X_i, LX_i, x_i, $A(X_i)$ >. Here X_i is the symbolic name of a linguistic variable (for example angular velocity or temperature), LX_i stands for the set of linguistic values that X_i can take on (Negative Big, Negative Medium, Positive Small etc.). The x_i is the actual physical range (the range of the crisp input) over which the

meaning of the linguistic values (angular velocity between –10 and 10 rad/s) can be determined. A(Xi) is a semantic function which maps elements of LX to a fuzzy set (membership function of a fuzzy set):

$$A(X_i): LX_i \rightarrow \mu_{LX}(x)$$

That is, A(Xi) is a function which takes a linguistic value as its argument (e.g. Negative Big) and returns the fuzzy set (membership function) associated with it.

4.2.3 Fuzzy Controller Design

This section provides several additional fuzzy related definitions necessary for the controller synthesis followed by a four-step design approach for realizing the fuzzy controller.

4.2.3.1 Definitions

Definition 4.2 *Fuzzy controller*

A fuzzy controller is a control system where the control signal is derived from a fuzzy rule-based expert system. The expert system utilizes a logical model of the thinking process a person might go through in the course of manipulating the system.

Definition 4.3 *Fuzzy engine*

The heart of the fuzzy controller – the fuzzy engine is an "if-then" rules based expert system. The "if-then" rule base is the logical model of the system. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic.

A single fuzzy if-then rule assumes the form:

IF x is A AND y is B then u is C,

where A, B and C are linguistic values defined by fuzzy sets on the ranges (universes of discourse) X , Y and U, respectively. The if-part of the rule "x is A" is called the antecedent, whereas the then-part of the rule "y is B" is called the consequent or conclusion.

Definition 4.4 *Degree of Applicability* [44]

The degree of applicability is the degree to which the whole antecedent part satisfies the rule. Usually it is denoted as β and is calculated as:

$$\beta = A_1(x_1) \, {}^t \, A_2(x_2) \, {}^t \ldots {}^t \, A_n(x_n),$$

where x_i is the value for the *i*-th crisp input, $A_i(x_i)$ is the membership degree for that value in the fuzzy set for the corresponding linguistic value and *t* denotes a *t*-norm operation. The *t*- norm can be calculated in different ways depending on the application. Most commonly *t* is calculated as minimum for AND fuzzy inference:

$$\beta = \min(A_1(x_1),\ A_2(x_2)\ \ldots,\ A_n(x_n))$$

and as maximum for fuzzy inference OR

$$\beta = \max(A_1(x_1),\ A_2(x_2)\ \ldots,\ A_n(x_n))$$

Output fuzzy set clipping and scaling

The degree of applicability β calculated from the antecedent part of the rule is used to modify the fuzzy set associated with the corresponding linguistic value describing the control output. Two techniques are used: clipping and scaling [44].

In the case of clipping (most commonly used) the height of the output fuzzy set is equal to the degree of applicability. That is, the membership function in the consequent part is cut on the level β.

If scaling is to be applied the membership function in the consequent part is factored by the degree of applicability. It means that all the membership degrees in the output are multiplied by β.

4.2.3.2 Controller Design Stages

This subsection presents a four-step design procedure for building a fuzzy controller:

Step 1, *Fuzzification*

The fuzzification process includes creating a linguistic model of the system to be controlled. The linguistic model is built as a set of linguistic variables (defined above). A linguistic variable is assigned for each of the observed (state) variables of the system as well as for the system's input. The number of linguistic values (LX_i in the linguistic variable definition) is usually between 3 and 7. Determined is the crisp range for each of

the linguistic values describing a state variable and a fuzzy set (membership function) is built for it.

Step 2 *Building the rule base*

This step includes creating the "if-then" based expert system as described above. The expert system utilizes the linguistic model of the system built in Step 1. The antecedent part of each rule includes one or more linguistic values. If the number of linguistic values is greater than one, they are related with AND, OR and NOT fuzzy expressions. According to the system to be controlled and the available computational power a decision has to be made whether a Mamdani or a Sugeno type fuzzy processing is to be used for the consequent part of a rule. Every rule in the rule base generates a fuzzy value (or a single spike in the Sugeno case) which is a "recommendation" for some control action.

Step 3 *Defuzzification*

The defuzzication step is required in order to produce a crisp value for the control output from the recommendations suggested from the rule base.

In the case of Mamdani-type fuzzy controllers a crisp control output is derived from the clipped (scaled) fuzzy sets proposed by the rule base. Depending on the application different defuzzification methods can be used but the two most common methods are the *centroid* and the *mean- of-maxima.*

In the discrete case (when there are some discrete values of the output signal with their membership degrees in the corresponding clipped/scaled fuzzy sets) the result of applying the centroid method is:

$$u = \frac{\sum_{i=1}^{k} u_i * \mu(u_i)}{\sum_{i=1}^{k} \mu(u_i)}$$

In the case of Sugeno-type fuzzy controllers the crisp control output is calculated as a weighted average of the singletons suggested from the rule base.

Step 4 *Tuning,* Tuning the fuzzy controller is the process of modifying input and output membership functions, by increasing or decreasing overlap, to make the controller work.

4.3 Sugeno Fuzzy Modeling for the Cart Pendulum System

The description of the strategy in this subsection is done with reference to the cart-pendulum system illustrated in Figure 4.2. The strategy is based on human operator behavior whose aim is to appropriately add energy to the pendulum in an effort to swing it to its inverted position.

As shown in [3] most energy can be added to the pendulum if the cart accelerates when the pendulum passes through its hanging down position (θ=0°) and when the pendulum approaches its inverted position. Then the strategy of a human operator who attempts to swing up the pendulum can be defined as follows. With 2-3 well-timed shakings of the cart the operator initially excites the pendulum. Then he/she waits during the pendulum's

swinging down. When the pendulum passes through its down position the operator applies force to accelerate the cart in the positive direction (This is done in the case when the pendulum has reached the down position after a swing down in a clockwise direction. If the pendulum swings down in a counterclockwise direction the cart accelerates in the negative direction). The magnitude of the applied force (respectively the magnitude of cart acceleration) should be conformable to the pendulum angular velocity. Thus if the angular velocity is big the cart must be accelerated less to provide some smoothness in the pendulum's advance to its inverted position. The operator keeps applying a force which decreases in magnitude as the pendulum approaches its horizontal position (θ=90). During the pendulum swinging in the neighborhood of its horizontal position the cart is kept motionless. When the pendulum approaches its inverted position the operator makes the decision for moving the cart based on the angular velocity of the pendulum. If the pendulum angular velocity is small then the operator adds energy to the pendulum by applying force to accelerate the cart in negative (positive) direction. If the angular velocity is large then the operator assumes that the pendulum will reach its inverted position and keeps the cart motionless.

Finally, when applying a given force the operator must make sure that the cart will not move out of the imposed cart rail limits.

As the force to the cart is directly related to the input voltage to pendulum apparatus, the input voltage will be considered as the output designed with the fuzzy engine.

.

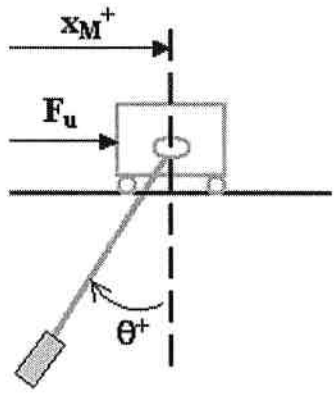


Figure 4.2 Pendulum on Cart system for Fuzzy Swing-up Control Modeling

Step 1 *Fuzzification*

The swing-up pendulum strategy as defined above shows the need for the pendulum angle, pendulum angular velocity and cart position to be selected as an input space. This input space has to be mapped via the fuzzy engine to the output space – the force applied to the cart. As the voltage supplied to the motor linear amplifier from the control computer through the DAC is related to the force applied on the cart (Equation A.4). It is most natural to consider directly this voltage as the output (control signal) from the fuzzy controller. The inputs and output for the fuzzy engine are defined in terms of linguistic variables, i.e. $< X_i, LX_i, x_i, A(X_i) >$. Linguistic variables are defined for the pendulum angle θ (*Pendulum Angle*), cart position x_M (*Cart Position*), pendulum angular velocity (*Angular Velocity*), and control signal u (*Control Signal*) and are presented in tabular form below:

Linguistic Name	Linguistic Values	Physical Range of the Crisp Input
Pendulum Angle	NVB, NB,NM, NS, PS, PM, PB, PVB	-180° ÷180°
Angular Velocity	NB, NM, NS, PS, PM, PB	-14rad/s ÷ 14rad/s
Cart Position	NB, NM, NS, PS, PM, PB	-0.18m ÷ 0.18m
Control Signal	NB, NM, NS, Z, PS, PM, PB	-6V ÷6V

NVB – Negative Very Big, NB – Negative Big, NM – Negative Medium, NS – Negative Small, ZN – Zero Negative, Z – Zero, ZP – Zero Positive, PS – Positive Small, PM – Positive Medium, PB – Positive Big, PVB- Positive Very Big

Table 4.1 Linguistic Variables for Pendulum Fuzzy Swing-Up Controller

(The position of the cart (crisp input) is measured with reference to the point at which the cart motion begins after the pendulum has passed through the position $\theta = 0°$)

Pictured in Figure 4.3-4.6 are the final forms of the fuzzy membership functions chosen for the respective linguistic values of *Pendulum Angle, Angular Velocity,* and *Cart Position*

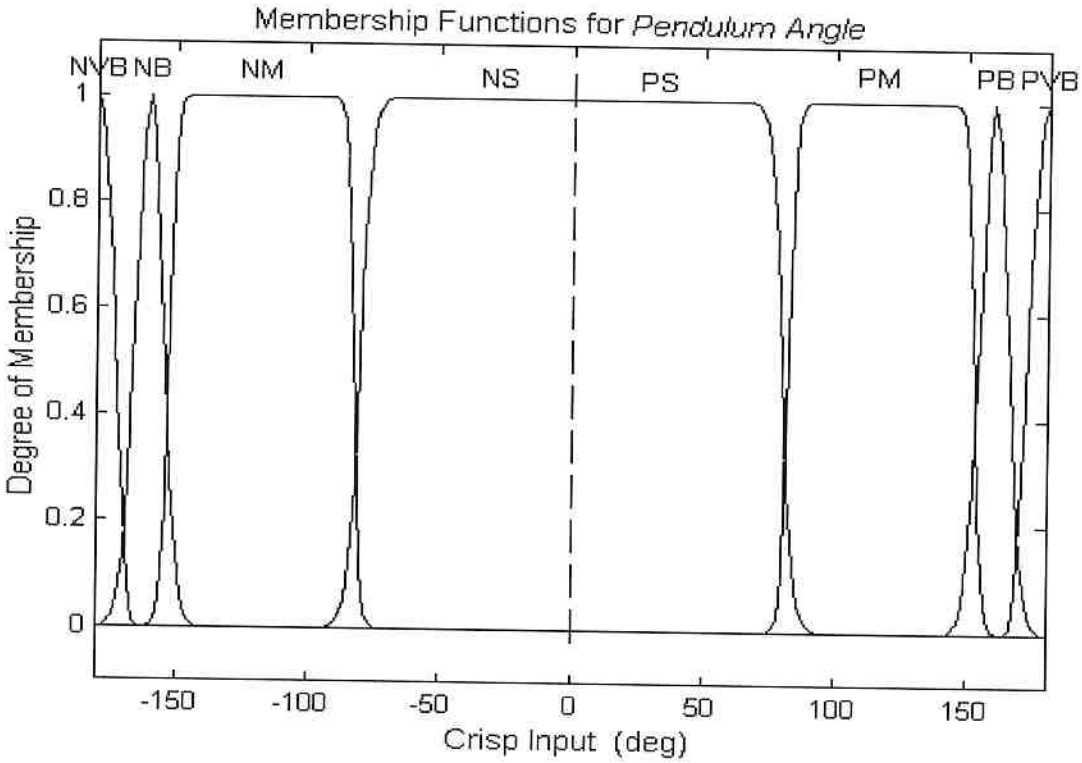


Figure 4.3 Membership Functions for *Pendulum Angle*

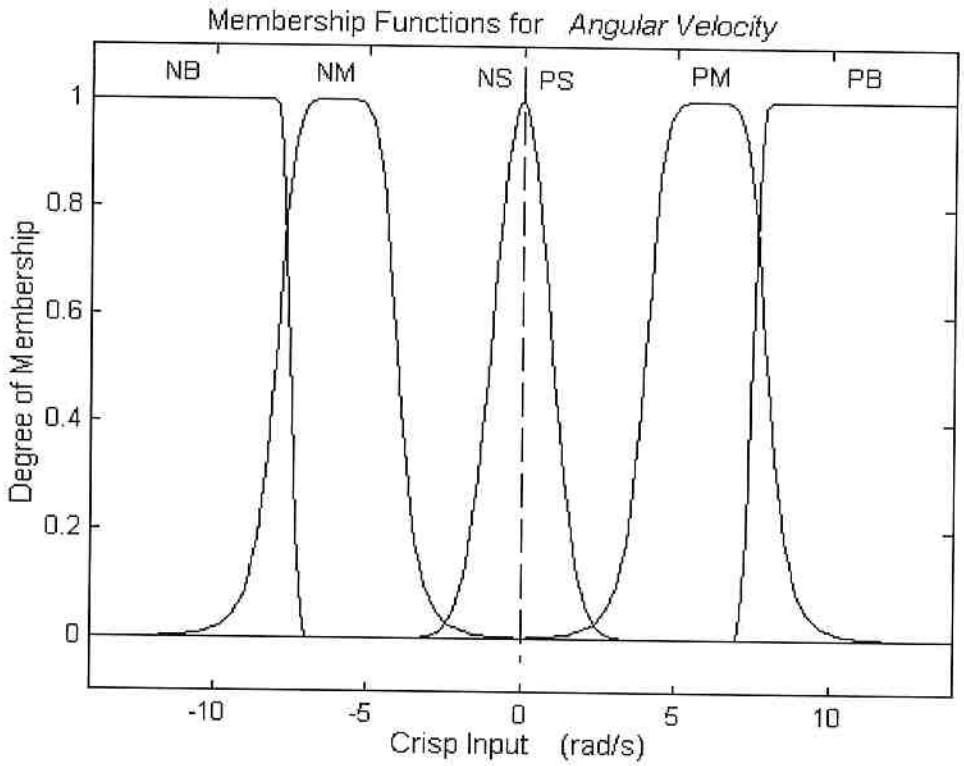


Figure 4.4 Membership Functions for *Angular Velocity*

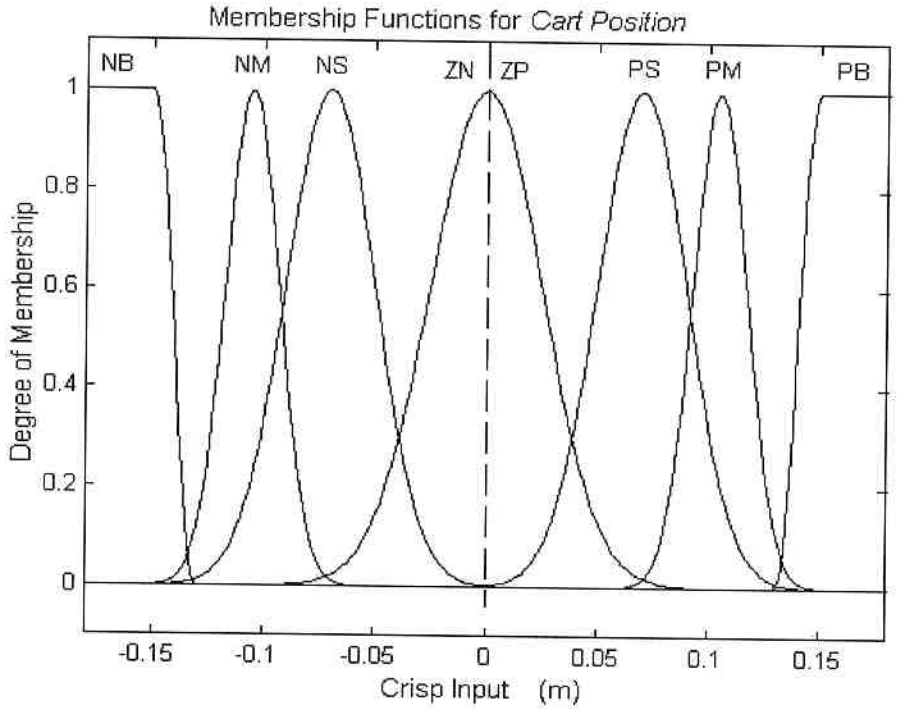


Figure 4.5 Membership Functions for *Cart Position*

The membership functions for *Control Signal* are defined as singletons to conform to the requirements of the Suguno fuzzy model.

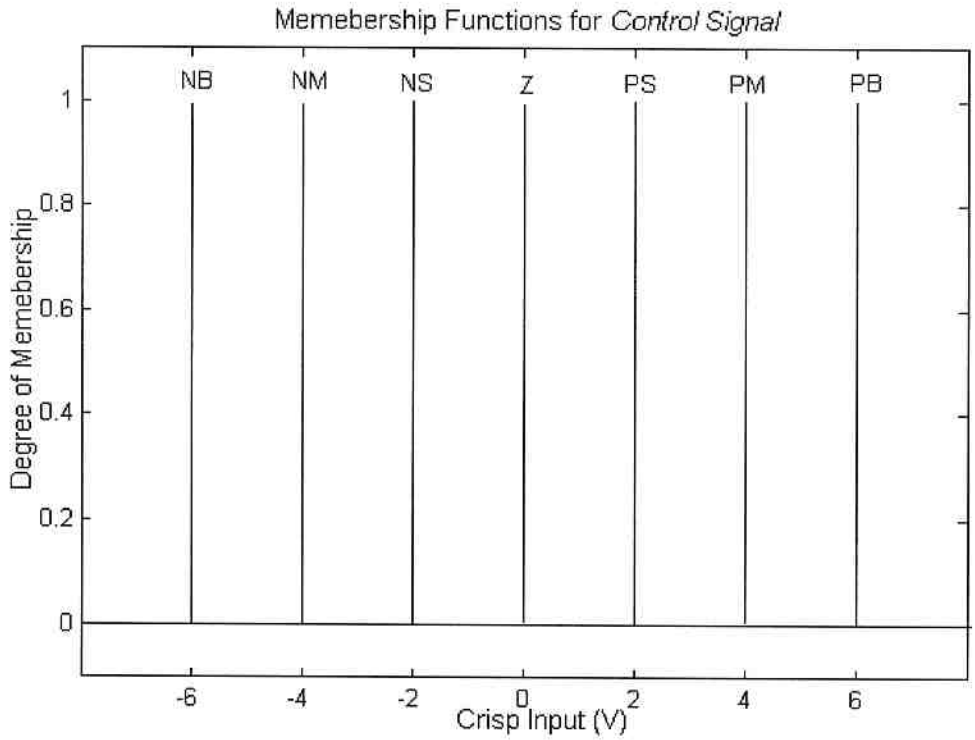


Figure 4.6 Membership Functions (singletons) for *Control Signal*

Step 2 *Building the rule base*

Step 2 is gives the rule base for the fuzzy engine implementing the human-operator based swinging strategy as described above. The case when pendulum swings down (i.e., $\theta\dot{\theta} < 0$) produces directly a zero crisp value for the control signal without entering the fuzzy engine. This case can be included in the rule base but would result in an unnecessary increase in the number of rules (the number of rules necessary to describe this scenario is proportional to the number of linguistic values of the fuzzy engine inputs, i.e. *Pendulum Angle, Angular Velocity, Cart Position*).

1. **IF** *Pendulum Angle* **is** PS **and** *Cart Position* **is** ZP **THEN** *Control Signal* is PB
2. **IF** *Pendulum Angle* **is** PS **and** *Cart Position* **is** PS **THEN** *Control Signal* is PM
3. **IF** *Pendulum Angle* is PS **and** *Cart Position* is PM **THEN** *Control Signal* is PS
4. **IF** *Pendulum Angle* is PS **and** *Cart Position* is PB **THEN** *Control Signal* is Z
5. **IF** *Pendulum Angle* **is** PS **and** *Angular Velocity* **is** PM **THEN** *Control Signal* is PM
6. **IF** *Pendulum Angle* is PS **and** *Angular Velocity* is PB **THEN** *Control Signal* is PS
7. **IF** *Pendulum Angle* is PM **THEN** *Control Signal* is Z
8. **IF** *Pendulum Angle* is PB **and** *Angular Velocity* is PS **THEN** *Control Signal* is NS
9. **IF** *Pendulum Angle* is PB **and** *Angular Velocity* is PM **THEN** *Control Signal* is Z
10. **IF** *Pendulum Angle* is PVB **THEN** *Control Signal* is Z

11. **IF** *Pendulum Angle* is NS **and** *Cart Position* is ZN **THEN** *Control Signal* is NB
12. **IF** *Pendulum Angle* is NS **and** *Cart Position* is NS **THEN** *Control Signal* is NM
13. **IF** *Pendulum Angle* is NS **and** *Cart Position* is NM **THEN** *Control Signal* is NB
14. **IF** *Pendulum Angle* is NS **and** *Cart Position* is NB **THEN** *Control Signal* is Z
15. **IF** *Pendulum Angle* is NS **and** *Angular Veloity* is NM **THEN** *Cntrol Signal* is NM
16. **IF** *Pendulum Angle* is NS **and** *Angular Velocity* is NB **THEN** *Control Signal* is NS
17. **IF** *Pendulum Angle* is NM **THEN** *Control Signal* is Z
18. **IF** *Pendulum Angle* is NB **and** *Angular Velocity* is NS **THEN** *Control Signal* is PS
19. **IF** *Pendulum Angle* is NB **and** *Angular Velocity* is NM **THEN** *Cntrol Signal* is PS
20. **IF** *Pendulum Angle* is NVB **THEN** *Control Signal* is Z

After closer examination of the rules it can be seen that antecedent part of the rules in the range 1-10 are symmetric to those from 11-20 with respect to the corresponding linguistic values. For example in rule 2 *Pendulum Angle*, *Cart Position* and *Control Signal* are PS,

PS and PM respectively and in rule 12 *Pendulum Angle*, *Cart Position* and *Control Signal* are NS, NS and NM. This situation is actually as expected since the rules are designed in a way that reflects the symmetric behavior of the cart-pole system when the pendulum swings up clockwise and counterclockwise, respectively. Thus, if the absolute values of the respective crisp inputs for the antecedent part of the rules are fed to the fuzzy engine the fuzzy set for the *Control Signal* can be determined just from the first 10 rules. Then with a small modification in the defuzzification step the crisp control value corresponding to the actual values of the input parameters can be calculated. Thus, the total number of the rules necessary to determine *Control Signal* is reduced to 10. This provides and an additional benefit. The membership functions for the fuzzy inputs as shown in Figures 4.2-4.4 may cause ambiguity in the determination of the membership degree for a crisp input of zero. Now, when absolute values of the crisp inputs are used, the membership function for the linguistic values PS and NS for *Pendulum Angle*, PS and NS for *Angular Velocity* and ZN and ZP for *Cart Position* can be combined to produce membership function for linguistic variables S (Small) for *Pendulum Angle*, S for *Angular Velocity* and Z (Zero) for *Cart Position* (the membership functions are built over generalized bell function in the case of *Pendulum Angle* and Gaussian distribution for *Angular Velocity* and *Cart Position* and are symmetric around crisp value of zero). This is realized in the fuzzy system built to simulate the behavior of the controller.

The rules are then given in a tabular form below:

Control Signal		*Cart Position*			
		Z	PS	PM	PB
Pendulum Angle	S	PB	PM	PS	Z
	PM	Z			
	PB				
	PVB	Z			

Control Signal		*Angular Velocity*		
		S	PM	PB
Pendulum Angle	S		PM	PS
	PM	Z		
	PB	NS	Z	
	PVB	Z		

Table 4.1 Fuzzy Rules

Step 3 *Defuzzification*

The fuzzy *Control Signal* recommended by the fuzzy engine is presented by scaled singletons (in the Sugeno type fuzzy processing)

A weighted average is performed over the scaled singletons to produce the crisp value of the control input *u:*

$$u = \frac{\sum_{i=1}^{n} \beta_i * k_i}{\sum_{i=1}^{n} \beta_i}$$

where n is the number of "fired" rules, β_i is the degree of applicability of the antecedent part of the ith "fired" rule and k_i is the crisp value at which appears the singleton for the linguistic value of *Control Signal* (e.g. PB) in the consequent part of the ith "fired" rule.

Since the absolute value for the crisp inputs are used by the fuzzy engine, the following simple modification with accordance to the sign of the current pendulum angle θ is necessary to form the final value of the control input:

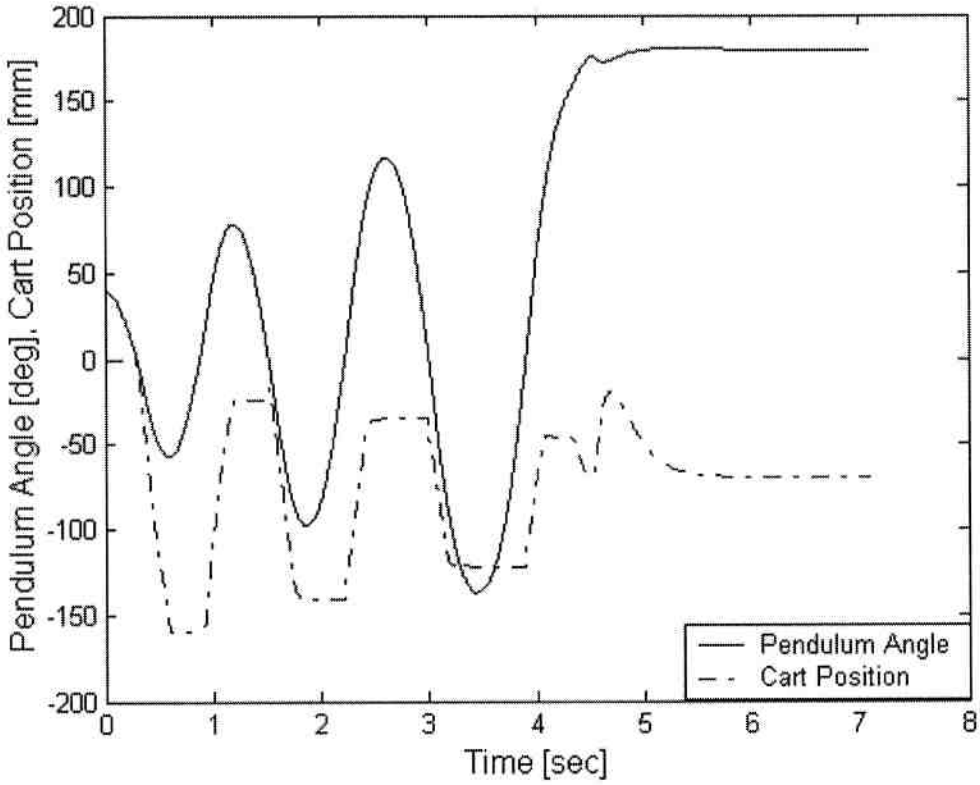
$$u = sign(\theta)u$$

Step 3 *Tuning*

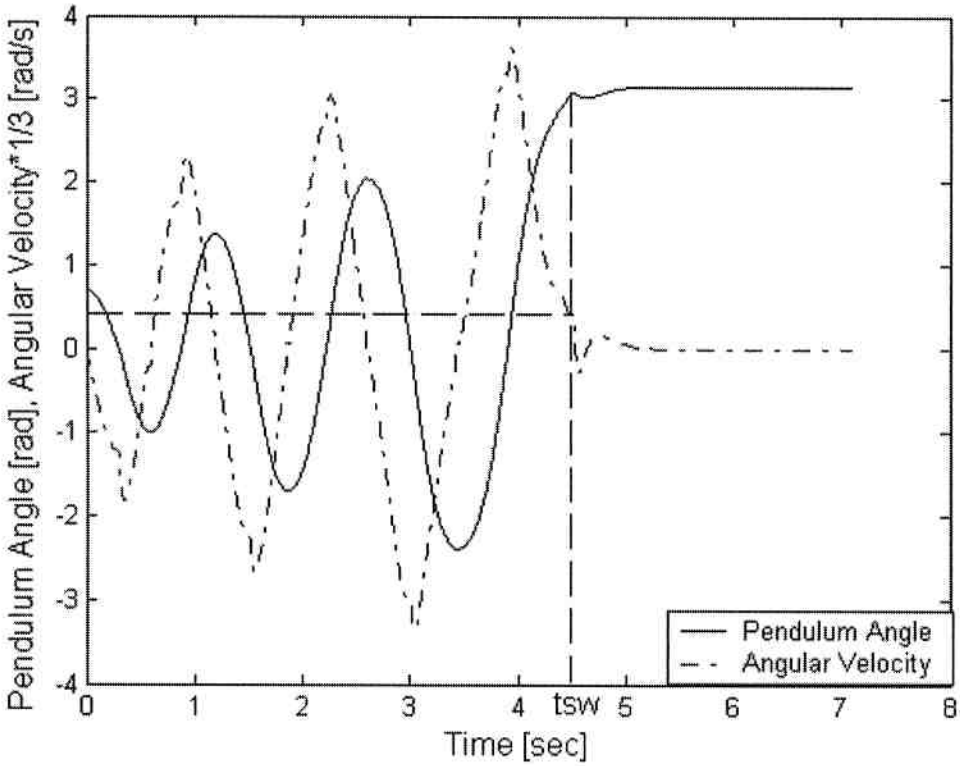
The final form of the membership functions are shown in Figures 4.2 – 4.5 and the rule base are a result of iterative process based on simulations. That is, the membership functions' form, overlap, size and position as well as the form of the rules has been iteratively altered to achieve the desired control performance.

4.4 Simulations

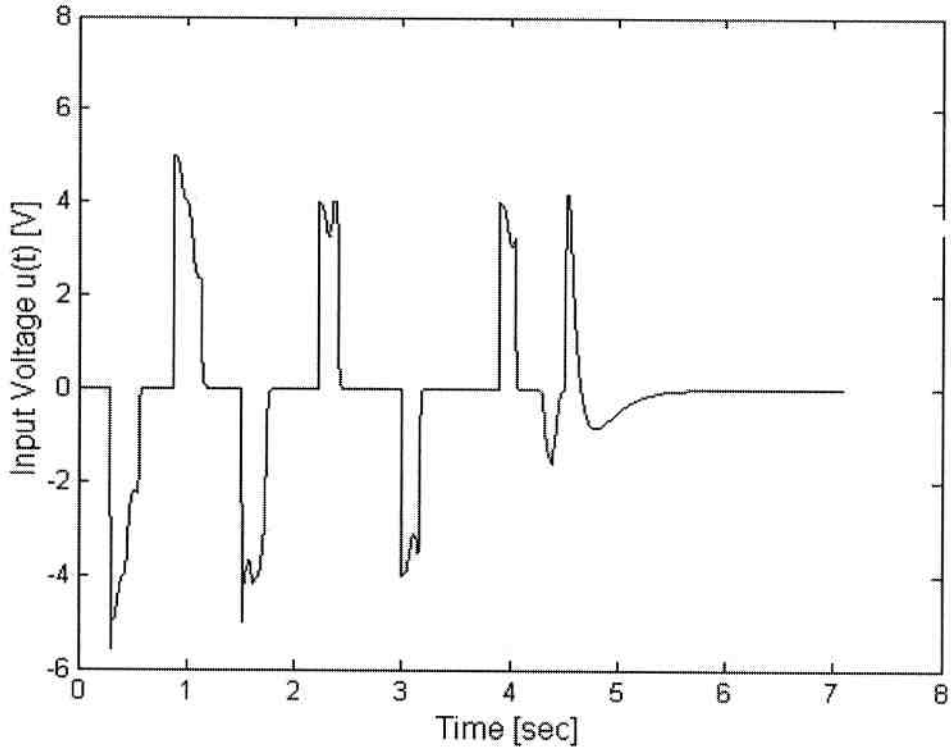
The fuzzy inference system has been built by utilizing the *Fuzzy Logic Toolbox* in Matlab®. Pendulum and cart movements are simulated numerically in the continuous time domain through Matlab's ODE solver. The respective Matlab® support scripts are given in Appendix D.



Graph 4.1 Cart/Pendulum System Response to Fuzzy Swing-up Controller and Linear State-Space Stabilizing Controller



Graph 4.2 Pendulum Angle vs. Pendulum Angular Velocity under Fuzzy Control



Graph 4.3 Control Signal Determined by the Fuzzy Engine for the Pendulum Swing-Up and Linear State-Space Control for the Stabilization

A successful fuzzy swing-up controller is considered to be the one which ensures that the pendulum smoothly approaches its inverted position. The small pendulum angular velocity will provide graceful switching to the controller that stabilizes the inverted pendulum (the linear state-space controller for the inverted pendulum stabilization is presented in Chapter 6). A smooth approach to the inverted position for the pendulum has turned out to be the main challenge during the tuning of the controller. As shown in Graph 4.2 at the moment of switching to the stabilizing controller the pendulum angular velocity is approximately 1.5 rad/sec. This value of angular velocity is in the acceptable range and it results in the short 4V peak in the control signal (Graph 4.3). Angular

velocities above 2 rad/sec at the switching time are undesirable since they would introduce higher demand to the actuator or even prevent successful stabilization of the inverted pendulum.

The simulation results from Graph 4.1 show the satisfactory behavior of the fuzzy pendulum swing-up controller and successful inverted pendulum stabilization. (The cart position at the moment of controller switching the control to the stabilization controller is considered as the zero cart position by the stabilization controller. Thus the cart position converges to this point)

4.5 Chapter Concluding Remarks and Recommendations

Fuzzy logic based controllers can be the appropriate choice for nonlinear control problems. Via simulations it has been shown that a fuzzy controller can be successfully designed for the pendulum swing-up problem.

In the experimental implementation, replacement of the *generalized bell* and *Gausssian* based membership functions with trapezoidal and triangular membership functions for the variables representing the input space can be considered. Although that by doing this some smoothness may be sacrificed, it is more effective in a computational sense.

There are reasons for not attempting experimental implementation of the controller. First, as seen from the simulations it is very difficult to achieve a smooth control signal and

thus to release the demand on the actuator. Second, there would be an inevitable difference between simulation and experimental results. As a result additional tuning of the fuzzy system will be required and repetitive failures of the tuning process on the physical system would possibly damage the equipment. Instead, it has been decided to proceed with the research for finding the most suitable way for swinging up the pendulum and to use the experience gained during the design of the fuzzy controller.

Chapter 5

Embedded Artificial Constraints Control

5.1 Chapter Introduction

The main consideration behind the design of the swing-up strategy presented in this chapter is the feasibility of implementation of the method on the physical apparatus.

The motivation behind initiating the research for this method was to attempt to transform the nonlinear pendulum swing up control problem into a linear controller design by introducing an artificial state link.

The algorithm is built by using an analysis which relates the pendulum energy to the cart acceleration. The method then prescribes a suitable, smooth, pendulum angle driven cart motion trajectory, which efficiently excites the pendulum towards its inverted position.

The chapter follows the logical path towards implementation of the swing-up control strategy and thus is organized in two parts: the first part of the chapter considers developing an algorithm for defining an appropriate cart trajectory and the second part presents the design of the cart position controller

5.2 System Equations of Motion

In the equations of motion of the cart pendulum system (B.18) derived in Appendix B, the pendulum angle θ is measured clockwise from the inverted position of the pendulum. For the analysis in this section it is more convenient to introduce a new coordinate [65]:

$$\hat{\theta} = \theta - \pi \tag{5.1}$$

From this point on in the analysis $\hat{\theta}$ will be written as θ, but it should be noted that θ now is measured clockwise from the down position of the pendulum as shown in Figure 5.1.

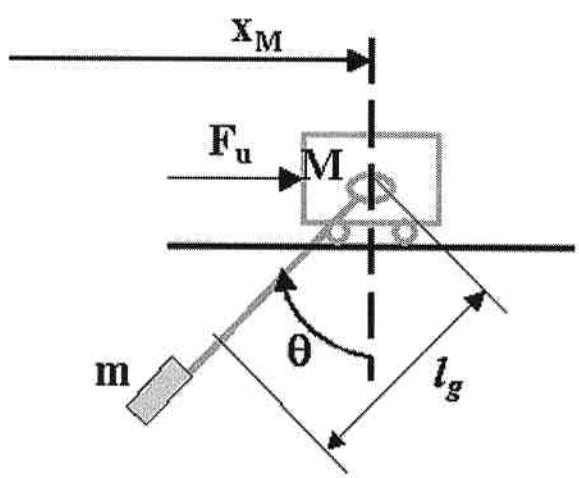


Figure 5.1 Pendulum on Cart system for EAC Modeling

Substituting the coordinate change (5.1) in the system dynamic equations (B.18) yields

the equation of the cart pendulum system when the pendulum angle θ is measured according to the direction shown in Figure 5.1:

$$\begin{cases} (M+m)\ddot{x}_M - ml_g \cos\theta\ddot{\theta} + ml_g\dot{\theta}^2 \sin\theta + b_c\dot{x}_M = \dfrac{A_G T_t}{r}u & (5.3) \\ \left(ml_g^2 + J_p\right)\ddot{\theta} - ml_g\ddot{x}_M \cos\theta + mgl_g \sin\theta + b_p\dot{\theta} = 0 & (5.4) \end{cases}$$

5.3 System State-Link Representation

If the entire pendulum system dynamics is considered, the pendulum angle and the cart position are coupled through the system differential equations (5.3) & (5.4). This sate coupling can be graphically described as shown in Figure 5.2.

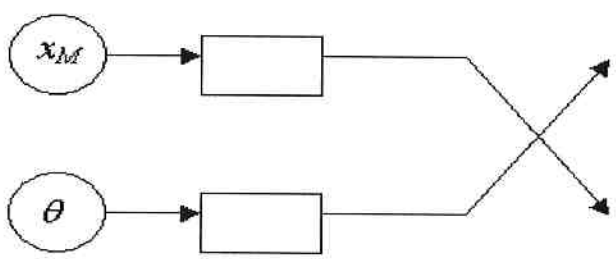


Figure 5.2 State Coupling

If a constraint for the cart motion based on the pendulum angle can be developed, such that when the cart moves with accordance with that constraint the pendulum is swung to its inverted position, then the relation between the pendulum position and cart position during the pendulum swinging-up could be presented just by a link:

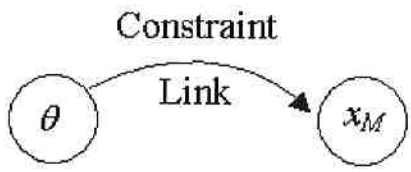


Figure 5.3 Dependence Between States presented by a Link

Consequently, the pendulum swing-up can be achieved by linearly controlling the cart and by observing the pendulum position without considering the entire system dynamics. The following subsections provide the procedure for derivation of such a constraint.

5.4 Artificial Cart-Motion Constraint Development

This subsection presents the derivation of an artificial link between the pendulum angle and the cart position.

5.4.1 Pendulum Energy Analysis

The design of a cart motion strategy which will lead to an increase of the pendulum's energy and raise it efficiently and predictably can be approached by analyzing the forces acting on the pendulum and the work done by these forces. A free – body diagram of the cart-pendulum system is shown in Figure 5.4.

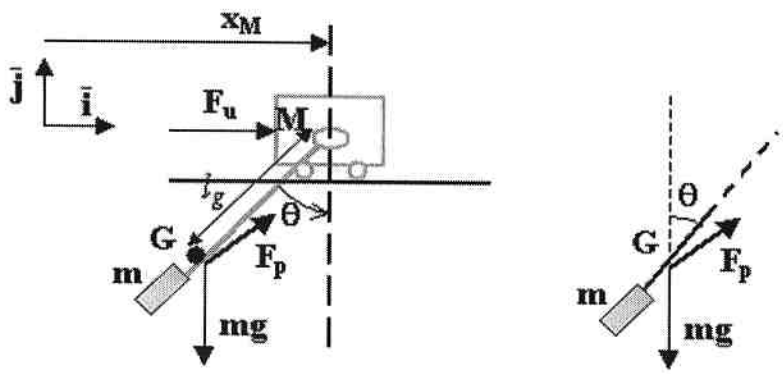


Figure 5.4 Pendulum on a Cart System Free Body Diagram

Two forces act on the pendulum center of mass: the force F_p and the force of gravity The force F_p can be presented as a sum of two component forces – Fr along the pendulum rod, and ΔF perpendicular to the F_r:

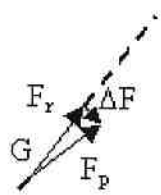


Figure 5.5 Components of the Force F_p Acting on the Pendulum Center of Mass

Since the pendulum mass moment of inertia has a very small value, J_p=0.0043 kgm^2, (which is due to the fact that the mass of the cylinder-rod pendulum is concentrated

mostly in the cylinder) the magnitude of the force ΔF, $|\Delta F|$, is very small and thus the force F_p can be approximated by its component, along the pendulum rod, F_r:

$$F_p \approx F_r \tag{5.5}$$

To simplify the following analysis it will be considered that the force F_p acting on the pendulum center of mass is along the pendulum rod, represented by F_r. This case is depicted in Figure 5.6

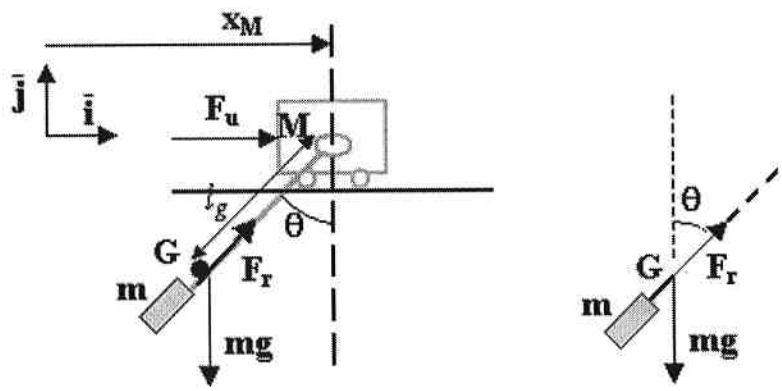


Figure 5.6 Pendulum on a Cart System Free Body Diagram with Approximated Forces

Using a Newtonian approach, expressions can be derived relating pendulum and cart motions. Using Cartesian coordinates, the motion and forces on the pendulum can be described. Throughout this analysis, bold characters are used to denote vector quantities, while non-bold characters represent scalar quantities.

The position of the pendulum center of mass can be derived from the cart position x_M and pendulum angle θ:

$$\mathbf{r} = (x_M - l_g \sin\theta)\hat{\mathbf{i}} - l_g \cos\theta\,\hat{\mathbf{j}} \tag{5.6}$$

The velocity and the acceleration of the pendulum center of mass can be found as the first and second time derivatives of its position:

$$\mathbf{v} = \frac{d\mathbf{r}}{dt} = (\dot{x}_M - l_g \cos\theta\dot{\theta})\hat{\mathbf{i}} + l_g \sin\theta\dot{\theta}\hat{\mathbf{j}} \tag{5.7}$$

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = (\ddot{x}_M + l_g \sin\theta\dot{\theta}^2 - l_g \cos\theta\ddot{\theta})\hat{\mathbf{i}} + (l_g \cos\theta\dot{\theta}^2 + l_g \sin\theta\ddot{\theta})\hat{\mathbf{j}} \tag{5.8}$$

As depicted in Figure 5.6, two forces act on the pendulum center of mass: the force **Fr**

$$\mathbf{F}_r = F_r(\sin\theta\,\hat{\mathbf{i}} + \cos\theta\,\hat{\mathbf{j}}) \tag{5.9}$$

and the force of gravity,

$$\mathbf{F}_g = -mg\hat{\mathbf{j}} \tag{5.10}$$

Applying Newton's Law,

$$\sum \mathbf{F} = m\mathbf{a} \tag{5.11}$$

leads to:

$$F_r \sin\theta\,\hat{\mathbf{i}} + (F_r \cos\theta - mg)\hat{\mathbf{j}} = \\ = m(\ddot{x}_M + l_g \sin\theta\dot{\theta}^2 - l\cos\theta\ddot{\theta})\hat{\mathbf{i}} + m(l_g \cos\theta\dot{\theta}^2 + l_g \sin\theta\ddot{\theta})\hat{\mathbf{j}} \tag{5.12}$$

This can be rewritten as two equations, one for each direction:

$$F_r \sin\theta = m(\ddot{x}_M + l_g \sin\theta\dot{\theta}^2 - l_g \cos\theta\ddot{\theta}) \tag{5.13}$$

$$F_r \cos\theta - mg = m(l_g \cos\theta\dot{\theta}^2 + l_g \sin\theta\ddot{\theta}) \tag{5.14}$$

F_r can be derived from equations (5.13) and (5.14) by eliminating $\ddot{\theta}$:

$$F_r = m(\sin\theta\ddot{x}_M + l_g\dot{\theta}^2 + g\cos\theta) \tag{5.15}$$

The aim of this analysis is to find efficient way for gradually adding energy to the pendulum, thereby swinging it up. The relation

$$\frac{dE}{dt} = \frac{dW}{dt} \tag{16}$$

shows that the change of the pendulum energy can be analyzed by analyzing the work transfer over the pendulum.

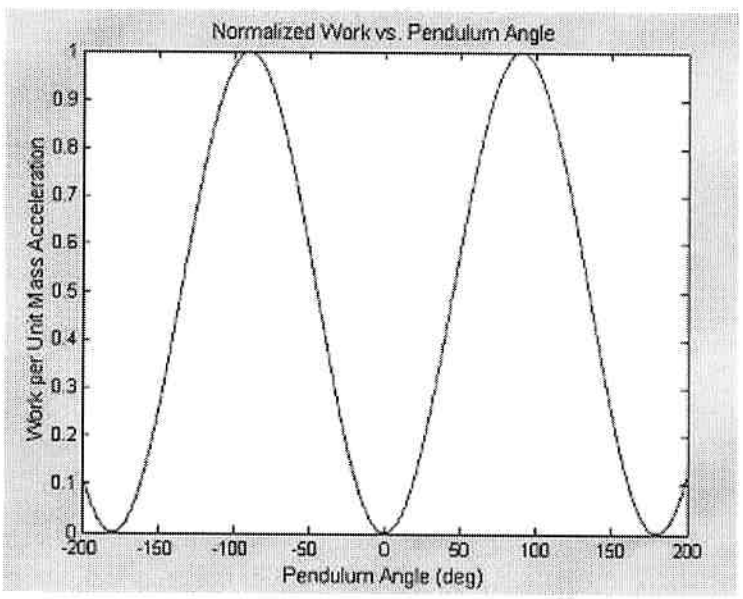
To find the work done on the pendulum by the force F_p (or its approximate F_r) when the cart moves a small distance, the component of the force that is in the direction of

movement is multiplied by the small horizontal component of the pendulum mass center displacement

$$\delta W = \mathbf{F_r} \bullet \delta r_i = F_r \sin\theta\, \delta r_i = m(\sin^2\theta\, \ddot{x} + l_g \sin\theta \dot{\theta}^2 + g \sin\theta \cos\theta)\delta r_i \quad (5.17)$$

Equation (5.17) shows three terms contributing to the pendulum force, and therefore contributing to the work done: an inertial force, a centripetal force, and a gravitational force. These components have their maximum work contributions at ±90°, ±90°, and ±45°, respectively.

The first term in Equation 5.17 is of great importance, as it describes the work effect of a given cart acceleration at any angle. Graph 5.1 shows a plot of this term.



Graph 5.1 Work/ Acceleration Ratio vs. Pendulum Angle

For θ near ±90°, a given acceleration and displacement will do a lot of work; for θ near 0°, hardly any work is done. As the case when the pendulum moves down is considered (i.e. the horizontal component of its displacement is in the same direction as the cart displacement) if the cart acceleration and displacement are in the same direction, the work done through the first term in (5.17) is positive and energy is added to the pendulum. If the acceleration and displacement are in opposite directions (deceleration), the work done is negative and energy is taken out of the pendulum.

5.4.2 Defining Cart Motion Strategy

As it desired that the cart be in motion just in the time range when the pendulum swings downwards, this range must include both cart acceleration and deceleration. Then, to maximize the positive work done on the pendulum, the trajectory should be such that the cart accelerates when there is high work transfer, i.e. θ near ±90° and decelerates when there is low work transfer, i.e. θ near 0° (it should be noted that cart deceleration when pendulum swings up is undesirable as it will do negative work, taking energy out of the pendulum).

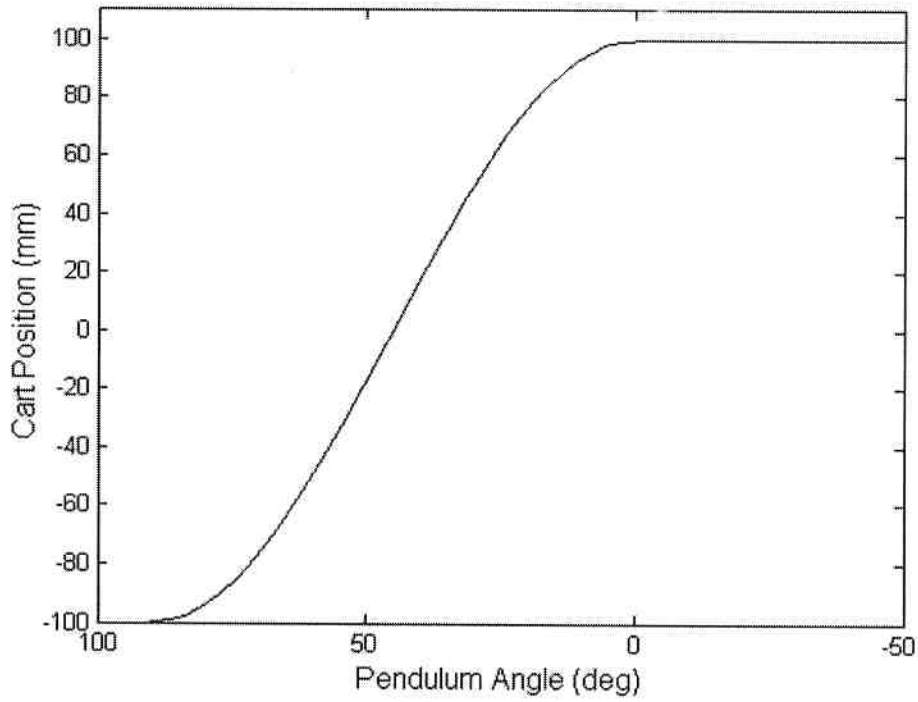
As the cart rail is of finite length, a cart position is specified instead of acceleration. The cart position trajectory should be prescribed in a way that provides cart acceleration and deceleration in the appropriate regions as described above.

A suitable, smooth trajectory for the cart position, directly driven by the pendulum angle θ, can be defined over half a period of a cosine wave:

$$x_M = x_{M_{ref}} + \mathrm{sgn}(\theta_{\max}) * (Amp) * \left(1 + \cos\left(\frac{\theta}{\theta_{\max}}\pi\right)\right), \qquad (5.18)$$
$$\theta \in [0, \theta_{\max}], \sin\theta\,\dot{\theta} < 0, \;\; Amp = Amp(\theta_{\max})$$

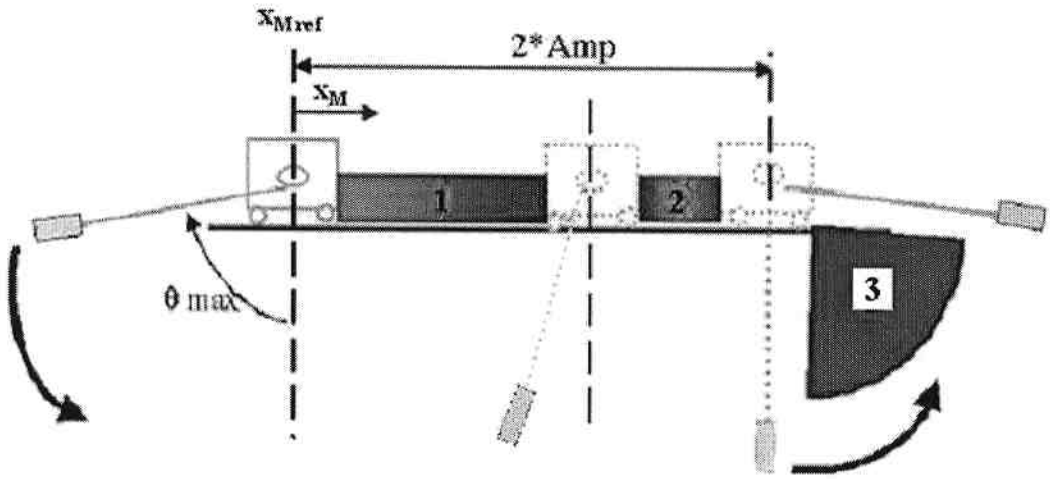
In Equation (5.18) $x_{M_{ref}}$ is the cart position at which each cart motion begins. The angle θ_{max} is the angle at which the swinging of the pendulum changes direction (from swinging upwards to swinging downwards). If the pendulum is not yet swinging above 90° (-90° for when the pendulum swings in the negative angle region), θ_{max} is determined to be the highest point of the swing (considerations for different selection of θ_{max} are given below, in the subsection describing the performed simulations). The term $\mathrm{sgn}(\theta_{\max})$ simply indicates at which direction the cart should move. The amplitude *Amp* defines the length that the cart must travel during the pendulum swinging downwards from $\theta = \theta_{max}$ to $\theta = 0°$. *Amp* is a function of the angle θ_{max}, i.e. $Amp=Amp(\theta_{max})$ and will be defined later.

The cart's displacement when the cart follows the trajectory given (5.18) and with pendulum released down from angle θ_{max} = 90° and with *Amp* = 0.1 m is given in Graph 5.2.



Graph 5.2 Cart Motion Over Pendulum Angle Driven Trajectory

Once the pendulum swings out of this region, to $\theta < 0$, the cart should remain motionless and wait for the pendulum to begin swinging downward again. Then a similar movement in the opposite direction should be prescribed.



1 – Cart accelerates 2 – Cart decelerates 3 – Cart is stationary

Figure 5.7 Proposed Cart Motion Strategy

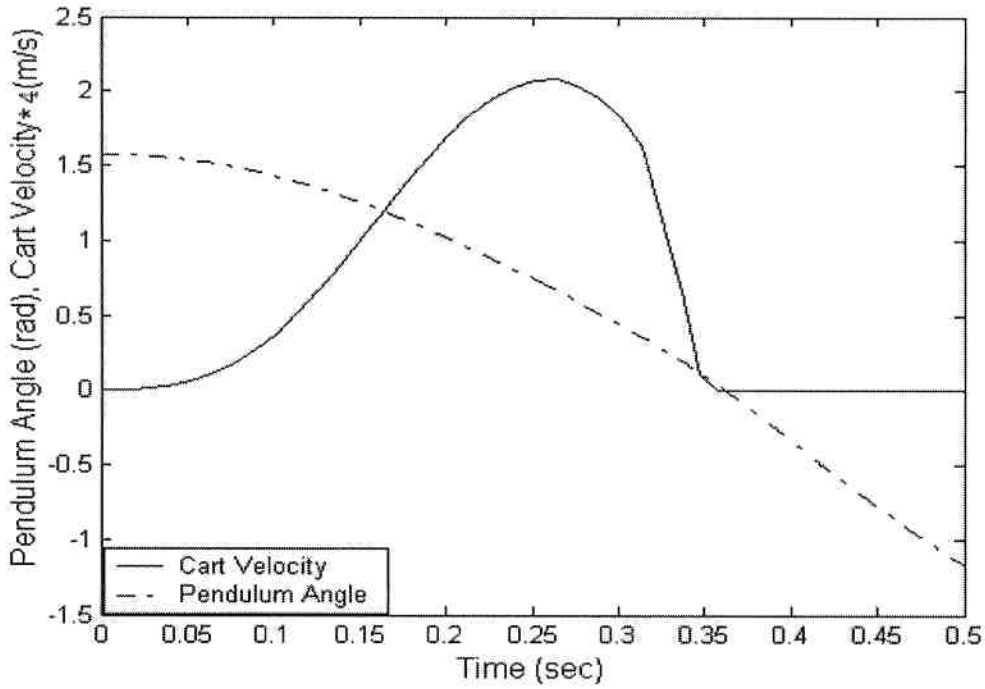
It is important to find the acceleration of the cart when this trajectory of movement is followed. Since the cart displacement as defined in Equation (5.18) is a function of θ, the cart velocity and acceleration will depend on derivatives of θ:

$$\dot{x}_M = \frac{\partial x_M}{\partial \theta}\dot{\theta} = -\frac{Amp\pi}{\theta_{\max}}\sin\left(\frac{\theta}{\theta_{\max}}\pi\right)\dot{\theta} \tag{5.19}$$

$$\ddot{x}_M = \frac{\partial \dot{x}_M}{\partial \theta}\dot{\theta} + \frac{\partial x}{\partial \dot{\theta}}\ddot{\theta} = -\frac{Amp\pi^2}{\theta_{\max}^2}\cos\left(\frac{\theta}{\theta_{\max}}\pi\right)\dot{\theta}^2 - \frac{A\pi}{\theta_{\max}}\sin\left(\frac{\theta}{\theta_{\max}}\pi\right)\ddot{\theta} \tag{5.20}$$

So the cart acceleration is not actually a cosine function, but involves the pendulum motion. When pendulum motion is considered, it can be shown that the cart acceleration and deceleration are in fact in the desired swing regions discussed earlier. This can be

seen and from the following Graph 5.3 which shows the velocity of the cart when the cart's displacement is the one shown in Graph 5.2:



Graph 5.3 Cart Velocity with the cart Following the Prescribed Trajectory

(The graphs presented in Graphs 5.2 and 5.3 are results from continuous time simulations using the Runge – Kutta method implemented in Matlab. The script is given in Appendix B, script file *cart_trajectory.m*)

For the trajectory described by equation (5.18), it is necessary that the pendulum indeed swings from $\pm\theta_{max}$ to 0°. Certain rapid cart movements will in fact cause the pendulum to accelerate upwards especially when the pendulum's swinging down begins at a low height. Special care must be taken during the simulations to assure the correct swinging

of the pendulum. The relation between the pendulum and cart acceleration is given by re-arranging Equation (5.4):

$$\ddot{\theta} = \frac{ml_g \cos\theta \, \ddot{x}_M}{ml_g^2 + J_p} + \frac{mgl_g \sin\theta + b_p\dot{\theta}}{ml_g^2 + J_p} \tag{5.21}$$

For the work analysis, the remaining terms in Equation (5.17) have not been considered. This is mainly because, unlike cart acceleration, $\dot{\theta}$ and gravity are not under direct control. Also, it is hoped that a fast controller will produce much greater effects through the first term.

It should be noted that the force approximation (5.5) indeed does not affect the results from the force work analysis. This is mainly because it was desired to derive the form of general trend of the effect that the cart acceleration has on the pendulum energy. If the force F_p was considered directly (instead of its approximate F_r) this trend would not be changed, i.e. the same general regions for cart acceleration/deceleration necessary for increasing the pendulum energy would be determined.

By imposing the artificial constraint (5.18) the method successfully establishes a link between the pendulum angle and cart position and transform the nonlinear control problem to linear cart position controller and eliminates the need of considering the entire system dynamics (5.3-5.4):

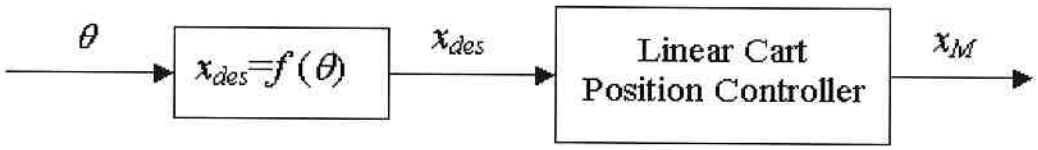


Figure 5.8 Linear Control by Imposing Artificial Constraint Equation

5.5 Cart Position Controller Design

The implementation of the swing-up controller as described in the previous subsections relies on an effective underlying cart position controller. This section outlines the design procedure for the controller. In this analysis the motion of the pendulum is considered as a disturbance to the system and therefore the controller is designed without taking into account the pendulum motion. The section begins with the derivation of the cart's equation of motion and the transfer function relating cart's position to the control input. After analyzing the behavior of the system the requirements for a controller are selected. The controller is implemented as a lead compensator and the behavior of the resulting closed-loop system is examined.

5.5.1 Cart's Equation of Motion and Transfer Function

When the pendulum swinging is ignored, the equation of motion of the cart is:

$$M\ddot{x}_M + b_M\dot{x} = F_u \tag{5.23}$$

where M and x_M are mass and horizontal displacement of the cart, b_m is the viscous friction coefficient and F_u is the force acting on the cart due to the actuator. According to the derivations in Appendix A the force F_u is related to the input control signal u as:

$$F_u = \frac{A_G T_t}{r} u \tag{5.24}$$

where A_G is the linear amplifier gain, T_t is the motor torque constant and r is the effective gear radius.

Substituting (5.24) in (5.23) and taking *Laplace* transform with zero initial conditions the cart's equation of motion in *Laplace* domain is

$$Ms^2 x_M(s) + b_M s x(s) = \frac{A_G T_t}{r} u(s) \tag{5.25}$$

Thus transfer relating the cart position to the control input is:

$$G_s = \frac{x(s)}{u(s)} = \frac{A_G T_t}{Mrs^2 + b_M rs} \tag{5.26}$$

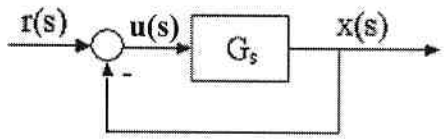
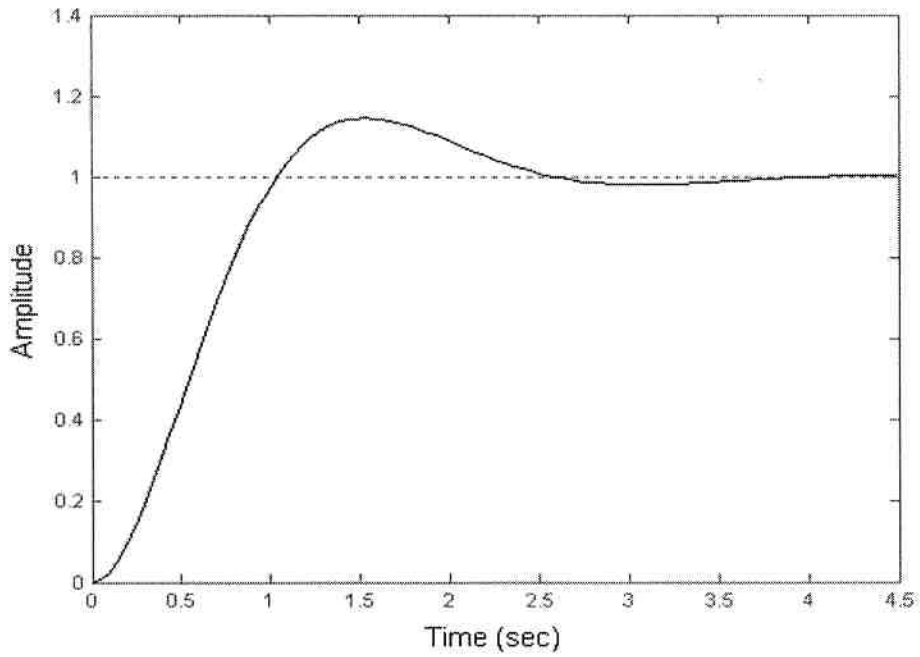


Figure 5.9 Cart System Closed-Loop Block Diagram

Examining the unit step transient response (Graph 5.4) of the closed-loop cart system with block diagram in Figure 5.9 shows an acceptable overshoot but the rise time and settling time have unacceptably large values (more than 1 and 2 seconds respectively), making the system's response too slow.



Graph 5.4 Closed-loop Cart System Transient Response

The transient response definitely shows necessity for implementing a controller in order for the time domain behavior of the system to be improved.

5.5.2 Design requirements

This subsection specifies the design requirements that have to be considered in the realization of the cart position controller.

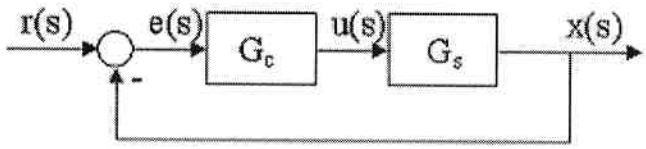


Figure 5.10 Controller/Cart system Closed –Loop Block Diagram

The closed-loop system including both the controller and cart transfer functions is given in Figure 5.10.

The goal is to achieve of a satisfactory transient response with a unit step input. It is required for the closed-loop system to not experience high overshoot as a significant overshoot causes unpredictability and the controller needs to be fast enough in order to reject the disturbance forces caused by the pendulum swinging. Thus the design requirements for the compensator are to produce a closed-loop system having a maximum percent overshoot of approximately 20% and settling time of approximately 0.1s:

$$M_p \approx 20\% \tag{5.27}$$

$$t_s \approx 0.1s \tag{5.28}$$

5.5.2.1 Compensated System Parameter Identification

The transient response to a unity step input of the closed-loop system shown in Graph 5.4 is governed by the dominant closed-loop poles and is given by [41]:

$$x(t) = 1 - \frac{e^{-\xi w_n t}}{\sqrt{1-\xi^2}} \sin\left(\omega_d t + \tan^{-1} \frac{\sqrt{1-\xi^2}}{\xi} \right) \tag{5.29}$$

where ξ, ω_n, ω_d with $\omega_d = \omega_n \sqrt{1-\xi^2}$ are the damping ratio, natural frequency and damped natural frequency of the system associated with the dominant closed-loop poles. The time t_p at which the maximum overshoot appears, can be found by differentiating (5.29) and letting the derivative equal zero:

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\xi^2}} \tag{5.30}$$

Substituting $t = t_p$ in (5.29) the maximum overshoot is found to be:

$$M_p = \exp\left(\frac{-\xi\pi}{\sqrt{1-\xi^2}} \right) \tag{5.31}$$

and the percent overshoot is:

$$M_p = 100 \exp\left(\frac{-\xi\pi}{\sqrt{1-\xi^2}} \right) \tag{5.32}$$

The settling time, measured when the transient response is within 5% of its steady-state value is:

$$t_s = \frac{3}{\xi\omega_n} \tag{5.33}$$

By noting that the additional non-dominant pole and zero will slightly affect the dynamic behavior of the closed-loop system governed by the dominant poles, the value for the maximum overshoot (5.32) associated with the dominant poles, should be chosen to be a bit below that required as by (5.27). Thus the design requirements for the dominant poles are selected as:

$$M_p = 15\% \tag{5.34}$$

$$t_s = 0.1s \tag{5.35}$$

Combining (5.34) with (5.32) and (5.35) with (5.33):

$$\begin{cases} 100\exp\left(\dfrac{-\xi\pi}{\sqrt{1-\xi^2}}\right) \approx 15\% \\ \dfrac{3}{\xi\omega_n} \approx 0.1s \end{cases} \tag{5.36}$$

From (5.36) the damping ratio and the natural frequency of the closed-loop system can be calculated as:

$$\begin{cases} \xi = 0.52 \\ \omega_n = 56 rad/s \end{cases} \tag{5.37}$$

5.5.2.2 Dominant Pole Locations

The damping ratio ξ as found above corresponds to an under-damped system and thus the dominant closed-loop poles are a complex conjugate pair.

The damping ratio together with the natural frequency ω_n for the desired closed-loop dynamic system calculated in (5.37) provide sufficient information for determining the exact location of the dominant poles in the s-plane. Let the complex conjugate poles be

$$(r_c, r_c^*) = a \pm jb \tag{5.38}$$

The characteristic equation associated with the dominant poles is:

$$s^2 + 2\xi\omega_n s + \omega_n^2 = 0 \tag{5.39}$$

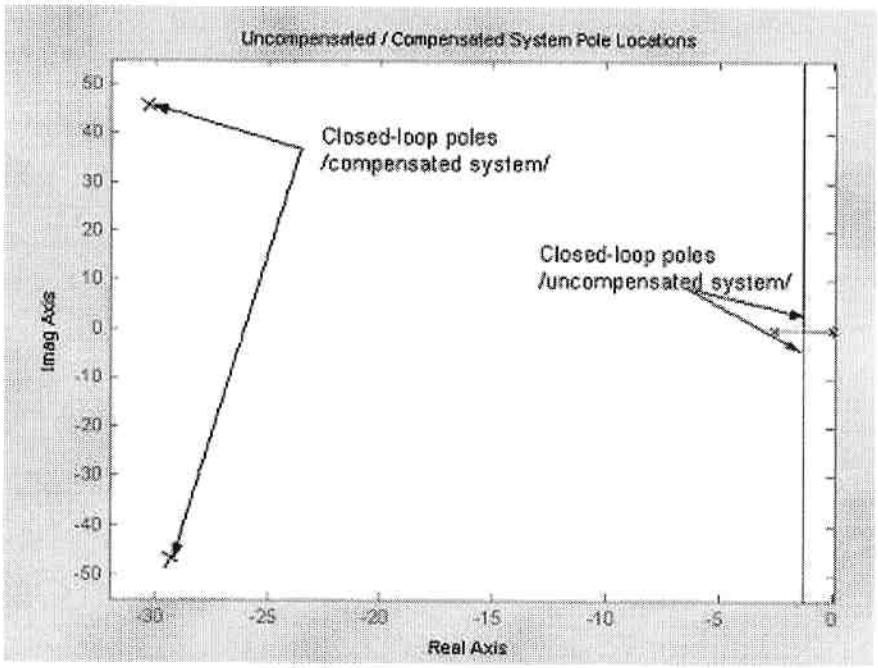
with roots

$$(r_c, r_c^*) = \omega_{\mathrm{n}}\left(-\xi \pm j\sqrt{1-\xi^2}\right) \tag{5.40}$$

By substituting (5.37) in (5.40) and comparing with (5.38) the desired closed-loop dominant poles are found to be:

$$(r_c, r_c^*) = -29.12 \pm j47.81 \tag{5.41}$$

Graph 5.5 shows the closed-loop poles locations for both the uncompensated cart system and the desired locations for the closed-loop poles for the compensated system:



Graph 5.5 Closed-Loop Pole Locations of the Cart System vs. Desired Pole Locations

5.5.3 Root-Locus Design

The controller G_c shown in Figure 5.6 is implemented as a lead compensator of the form:

$$G_c = K_c \frac{s+\tau_1}{s+\tau_2} \ , \ \tau_1 << \tau_2 \tag{5.42}$$

The compensator design is done in time domain by utilizing the root-locus methods. The overall closed-loop transfer function for the controller-cart system as shown in Figure 5.6 is:

$$TF_{CL} = \frac{G_c G_S}{1 + G_c G_S} \tag{5.43}$$

The characteristic equation of the system described by (5.43) is

$$1 + G_c G_S = 0 \tag{5.44}$$

or

$$G_c G_S = -1 \tag{5.45}$$

Thus, in order to force the root-locus of the closed loop-system to pass through a given point $s = s_0$ in the s plane the following conditions should be satisfied:

$$\arg(G_c * G_s)\Big|_{s=s_0} = \pm 180^\circ * (2k+1),\ k = 0,1,2,... \tag{5.46}$$

$$abs(G_c * G_s)\Big|_{s=s_0} = 1 \tag{5.47}$$

Now, following the requirement set in (5.41) the root locus of (5.43) should be forced to pass through the points s_1 and s_2 in the s plane presented by the complex conjugate

numbers $s_1 = -29.12 + j47.81$ and $s_2 = -29.12 - j47.81$. Since the root loci are symmetrical with respect the real axis, just $s_1 = -29.12 + j47.81$ will be considered and the same considerations given below for that point can be applied for $s_2 = -29.12 - j47.81$.

For the point $s_1 = -29.12 + j47.81$ the angle condition (5.46) implies:

$$\arg(G_c * G_s)\Big|_{s=-29.12+j47.81} = \pm 180^\circ * (2k+1),\ k = 0,1,2,... \qquad (5.48)$$

Since the angle of the cart system *Gs* at the desired closed-loop pole is:

$$\arg(G_s)\Big|_{s=-29.12+j47.81} = ang\left(\frac{0.139}{0.024s^2 + 0.061s}\right)\Bigg|_{s=-29.12+j47.81} = -241.43^\circ \qquad (5.49)$$

in order to force the root locus to go through the desired closed-loop pole, the lead compensator must contribute an additional angle of $\phi = 61.43^\circ$.

Thus if the zero of the lead compensator defined by (5.42) is chosen to be at s = -7.12 or $\tau_1 = 7.12$ (comments on this choice will be given below) the location of the pole of the compensator (respectively the value of τ_2) can be found from:

$$ang(G_c)\Big|_{s=-29.12+j47.81} = 61.43^\circ \qquad (5.50)$$

$$ang\left(K_c \frac{s+7.12}{s+\tau_2}\right)\Bigg|_{s=-29.12+j47.81} = 61.43^\circ \qquad (5.51)$$

therefore

$$\tau_2 = 64.27 \tag{5.52}$$

and thus:

$$G_c = K_c \frac{s + 7.12}{s + 64.27} \tag{5.53}$$

The compensator's gain can be determined from condition (5.47) for $s_0 = -29.12+j47.81$

$$abs(G_c * G_s)\big|_{s=-29.12+47.81} = 1 \tag{5.54}$$

$$\Updownarrow$$

$$abs\left(K_c \frac{s + 7.12}{s + 64.27} \frac{0.139}{0.024s^2 + 0.061s} \right)\Big|_{s=-29.12+47.81} = 1 \tag{5.55}$$

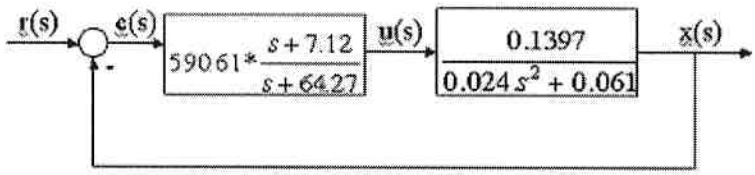
$$\Updownarrow$$

$$K_c = 590.61 \tag{5.56}$$

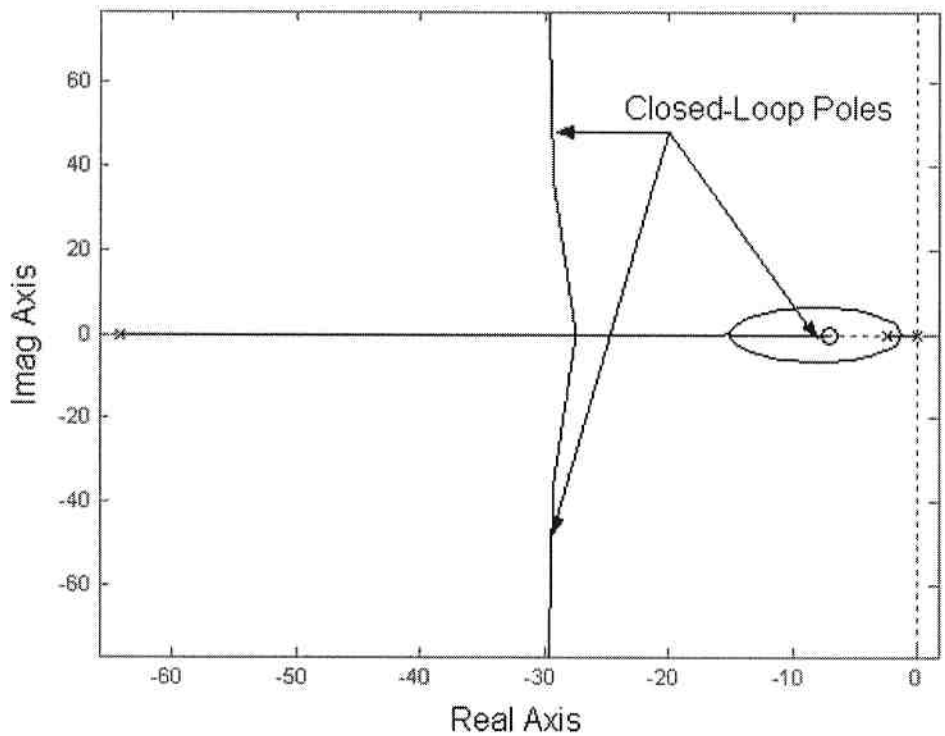
Finally, combining (5.53) and (5.56) the compensator for the cart system, which satisfies the requirements (.13) is found to be:

$$G_c = 590.61 \frac{s + 7.12}{s + 64.27} \tag{5.57}$$

Thus the overall closed-loop system combining the cart system and the position controller becomes:



The root-locus of the system is given in Graph 5.6

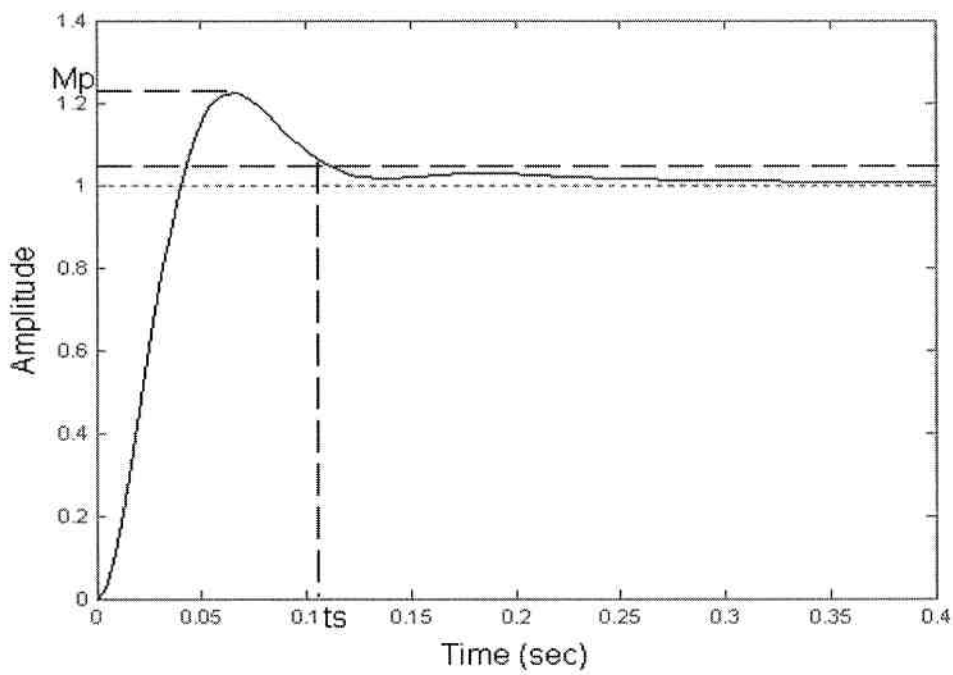


Graph 5.6 Compensated Cart System Root-Locus

Note that the root locus given in Graph 5.6 corresponds to a system:

$$System_{CL} = \frac{Gain * G_c G_S}{1 + Gain * G_c G_S} \tag{5.59}$$

where *Gain* varies from zero to infinity. When *Gain* equals one (5.39) becomes equal to (5.43) and the locations of the closed-loop poles of the system are marked on the graph. It is clear that the complex conjugate poles are exactly as specified by (5.41). The third closed-loop pole is at s = -7.75 and thus is very close to the zero of the system and consecuently they have very small effect on the system's behavior. The unit step transient response of the system is given in Graph 5.7



Graph 5.7 Compensated Cart System Unit Step Response

From figure 5.10 it is clear that the behavior of the system satisfies the requirements (5.27) and (5.28) for maximum overshoot and settling time.

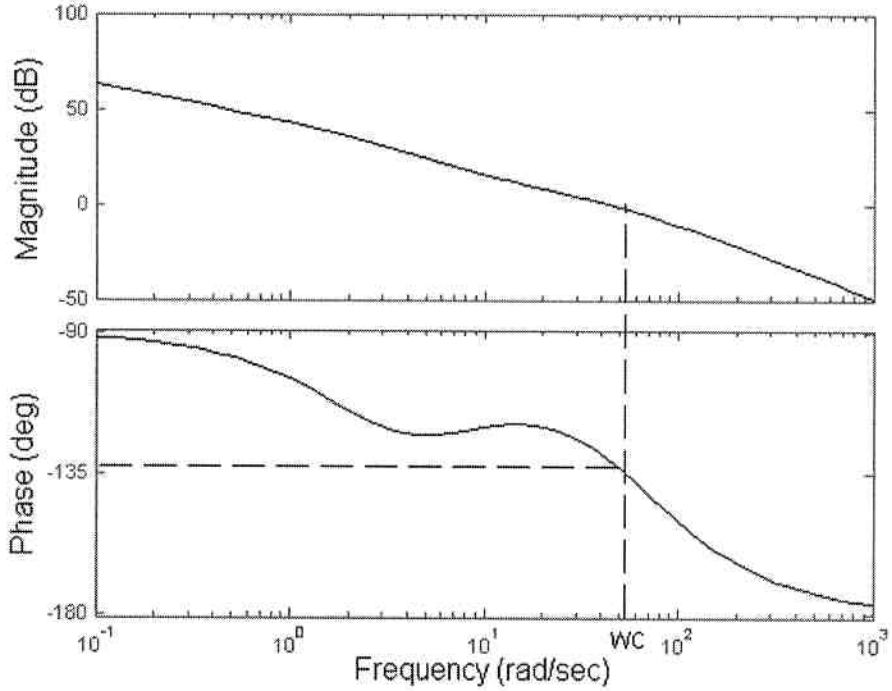
5.5.4 Frequency Domain Behavior

The compensator as designed by the root-locus method ensures satisfactory behavior of the system in the time domain. It is important to show that with that controller the system has an acceptable frequency characteristic. For stability purposes the gain and phase margins of the open-loop system must be examined. The bandwidth determined from the magnitude frequency response should be examined in order for an appropriate sampling rate to be selected for the digital equivalent of the compensator.

The open-loop transfer function for the controller-cart system as shown in Figure 5.7 is

$$TF_{OL} = G_c * G_s = 590.61\frac{s+7.12}{s+64.27} * \frac{0.139}{0.024s^2 + 0.061s} \tag{5.60}$$

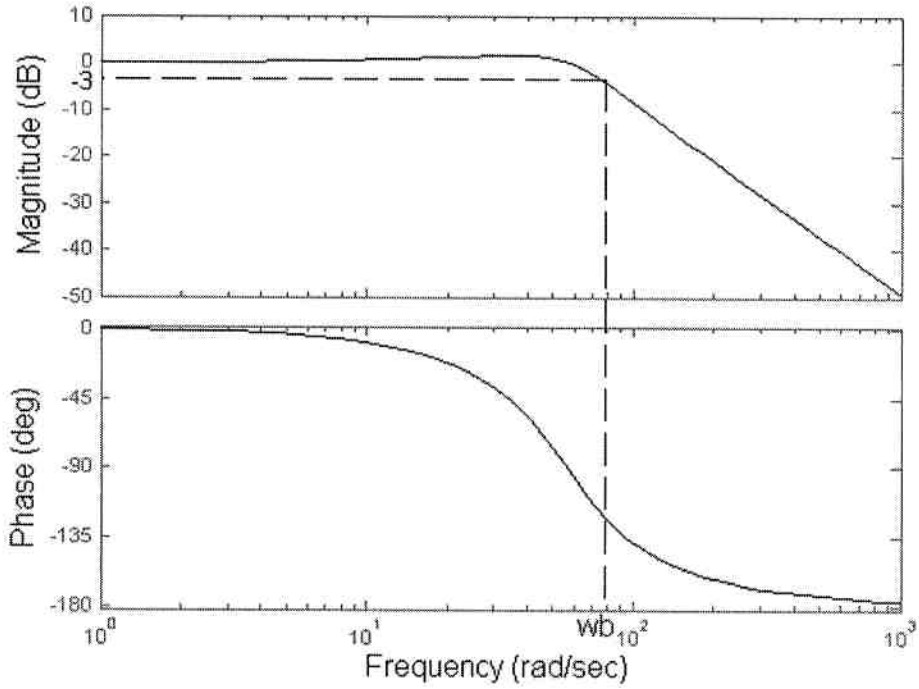
The Bode diagram presenting both the magnitude and phase frequency responses for the open loop system is given in Graph 5.8.



Graph 5.8 Compensated Cart System Open-Loop Frequency Response

From the graph it can be determined that the system's crossover frequency appears at ω_c=44 rad/sec and the phase margin of the system is γ=50° which is sufficient for stable operation.

The closed-loop frequency response of the system is given in Graph 5.9.



Graph 5.9 Compensated Cart System Closed-Loop Frequency Response

The bandwidth of the system, defined to be the frequency range up to the frequency at which the output is attenuated to a factor of 0.707 times the input (or down 3 dB) is found to be $\omega_b = 80 rad/\sec$

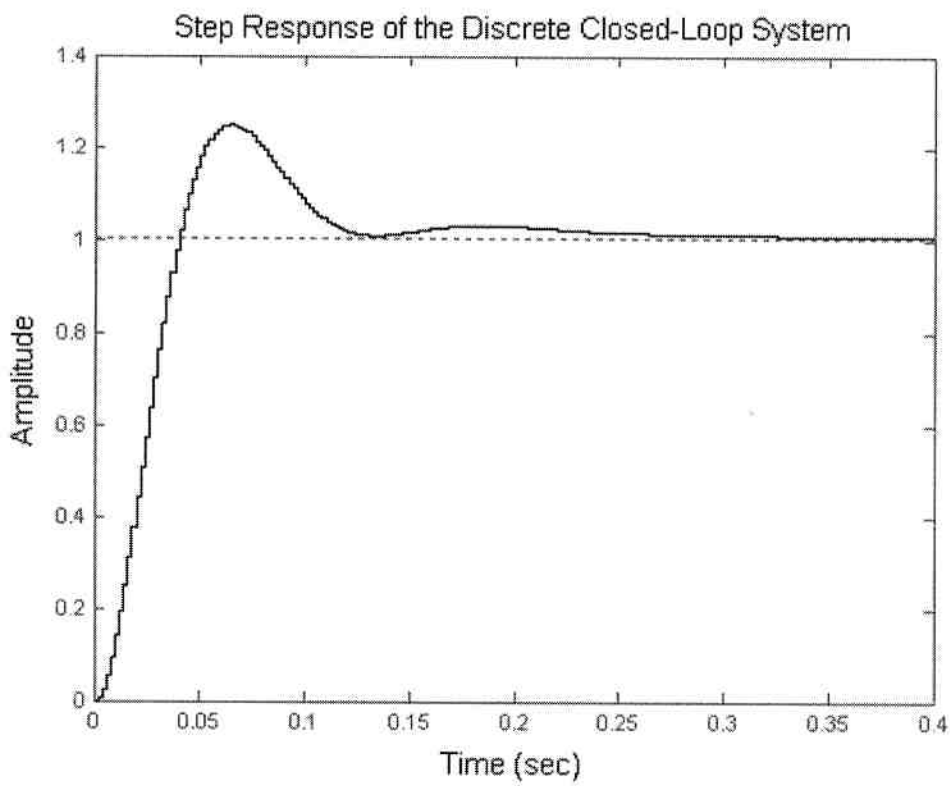
5.5.5 The Discrete Equivalent

In order the position controller to be implemented in a digital computer a discrete equivalent for the continuous time controller must be find. With reference to [15], suitable sampling rate for the discrete system should be up to 40 times the bandwidth of the system and thus the sampling rate is chosen to be 500Hz.

The discrete equivalent can be found by carrying the calculation manually or by using a CAD system. By utilizing the Matlab's function *c2d* which implements the Zero-Order Hold method, the discrete form of the compensator is found to be:

$$G_c^d = 590.61\frac{z-0.98}{z-0.8794} \tag{5.61}$$

The unit step response of the closed-loop discrete cart/controller system (found again with the *c2d* function) is given in Graph 5.10.



Graph 5.10 Discrete Closed-Loop System Unit Step Response

It can be noticed from Graph 5.10 that as expected, the step response of the discrete system has a very similar to the continuous system behavior. The overshoot is slightly increased when compared to the continuous time response in Figure 5.10. This is due to the fact that the zero and hold introduces additional phase shift at crossover equal to $\omega_c*T/2$ (where T is the sampling period), which decreases the system's phase margin and consequently the damping ratio which results in an increased overshoot.

5.5.6 Some Comments on the Compensator's Design

The design requirements

More strict design requirements than (5.27) and (5.28) can be set for the closed-loop system, i.e. smaller overshoot and shorter settling time can be specified. A lead compensator satisfying those requirements can be designed by following the same procedure as described above. The problem that arises is that the compensator's gain K_c is too large and thus unrealistic demands to the actuator will be imposed.

On the other hand when the requirements (5.27) and (5.28) are met the resulting system is somehow optimal in a sense of time and frequency domain characteristics and actuator demand.

The choice of the compensator's zero location

In the calculations leading to (5.53) the zero of the compensator was chosen to be at s = -7.12 and then by using the angle condition for the root-locus method, the pole was

found to be at s = -64.27. It has to be noted, that there are infinite number of zero/pole compensator's pair that satisfy the angle condition (5.50). The position of the zero was chosen very carefully after a number of simulations were held in order to provide the best locations for the closed-loop system poles and zero, which in turn satisfy the design requirements.

5.6 Swing-Up Controller Implementation

The swing-up controller is responsible for following tasks: to initiate the swinging of the pendulum, to add energy to the pendulum until it reaches its inverted position and to switch gracefully to the stabilization controller (the design of the state-space controller which stabilize the pendulum in the inverted position is describe in the next chapter).

The trajectory of the cart as defined by Equation (5.18) is ineffective when the pendulum is in rest (i.e. $\theta = 0°$) as the work transfer is zero. The pendulum can be excited until it reaches $\theta \sim 40°$ by several (2 or 3) rapid cart movements, which change direction when the pendulum crosses $\theta = 0°$. Then the cart is moved in accordance with the trajectory prescribed by (18) and thus energy is added to the pendulum. The amplitude *Amp* that the cart travel must be decreased as the pendulum swings high. This allows the inverted position ($\theta = \pm 180°$) to be reached smoothly. The amplitude function $Amp=Amp(\theta_{max})$ can be tuned to produce the desired result. More elaborate calculations could calculate the current energy of the pendulum, (equation for the pendulum energy can be derived from the analysis in Appendix B), and prescribe a trajectory that deliveres a certain amount of

work. However, this calculation is computationally intensive, as it involves a work integral with continually changing parameters.

The swing-up controller passes the control to the stabilization controller when the pendulum reaches the standing conditions. The switching conditions are determined as:

$$\begin{aligned} |\theta| &\geq 178° \\ |\dot{\theta}| &\leq 1\,\text{rad/s} \end{aligned} \tag{5.22}$$

If the pendulum approaches the vertical with angular velocity higher than 1 rad/s stabilization is not attempted. Rather than this, a recovery procedure is initiated. The recovery procedure waits the pendulum to start swinging down from $\theta_{max} = 40°$ and starts the swinging-up process again.

The stabilization controller is a linear state-space controller and is described in the next chapter.

5.7 Simulations

The simulations are performed for the discrete time case with sampling period of T=0.002sec. The cart position controller is implemented as per Equation (5.61). The Matlab script is given in Appendix D, script file *energy_swing_stab.m*. In the simulation every consequative system state is derived from the current state and current control input by using the difference equation:

$$\mathbf{x}(kT+T)=\mathbf{\Phi}\mathbf{x}(kT)+\mathbf{\Gamma}u(kT)$$

The matrices $\mathbf{\Phi}$and Γ are derived from the continuous time system through the Matlab function *c2d*(). The *c2d* function takes as an input the matrices **A** and **B** representing the continuous time linearized model about the current system state ($\dot{\mathbf{x}}=\mathbf{A}\mathbf{x}+\mathbf{B}u$). The linearization of the continuous system at each discrete time kT is done by following the approach presented in [61]

For the simulations, a table with several values of function/argument has been built for the function $Amp=Amp(\theta_{max})$ which determines the distance that the cart travels during pendulum swinging downwards. A linear interpolation between the entries in the table is then performed for each given value of θ_{max}.

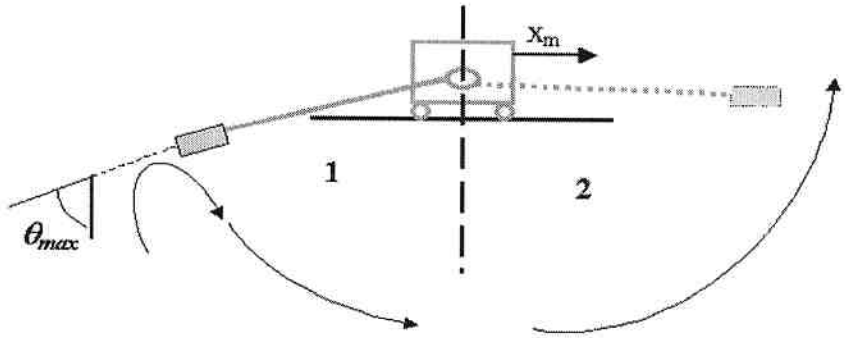
The entries in the table have been carefully selected in order to ensure that the pendulum reaches its inverted position with a angular velocity less than 1rad/sec which allows a smooth switching to the stabilizing controller. Since $Amp=Amp(\theta_{max})$ is an even function the table is built over absolute values of the argument θ_{max}.

abs(θ_{max}) (deg)	0	60	110	130	150	165	180
Amp (mm)	70	50	45	30	26.2	23	15

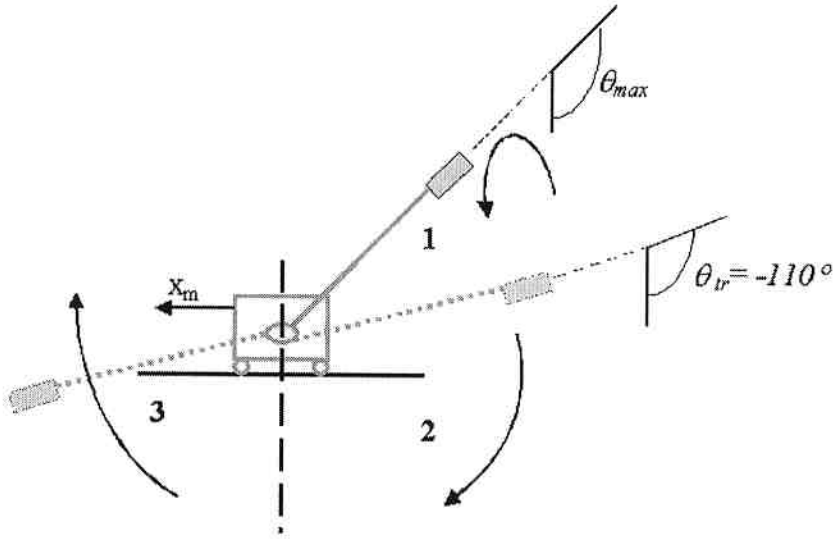
Table 5.1 Lookup table defining values for cart excursion amplitude as a function of θ_{max} (simulations)

The determination of θ_{max} depends on the highest point of swinging. The experience gained from several simulations leads to a selection of the threshold pendulum angle at which cart motion begins as θ_{tr}=110°. This value ensures that the cart acceleration best

fits the corresponding high work transfer region when the cart follows the prescribed trajectory. Thus if the pendulum is not yet swinging past 110° (-110° for when the pendulum swings in the negative angle region), θ_{max} is determined to be the highest point of the swing. Otherwise the amplitude that the cart must travel, *Amp*, is determined with accordance to the highest point of swing, i.e. $Amp=Amp(\theta_{max})$, and then θ_{max} is assigned the value of the threshold angle, i.e. $\theta_{max} = \theta_{tr}$ and the cart does not start moving until $\theta = \theta_{tr}$. The two different swinging scenarios are illustrated in Figure 5.11.



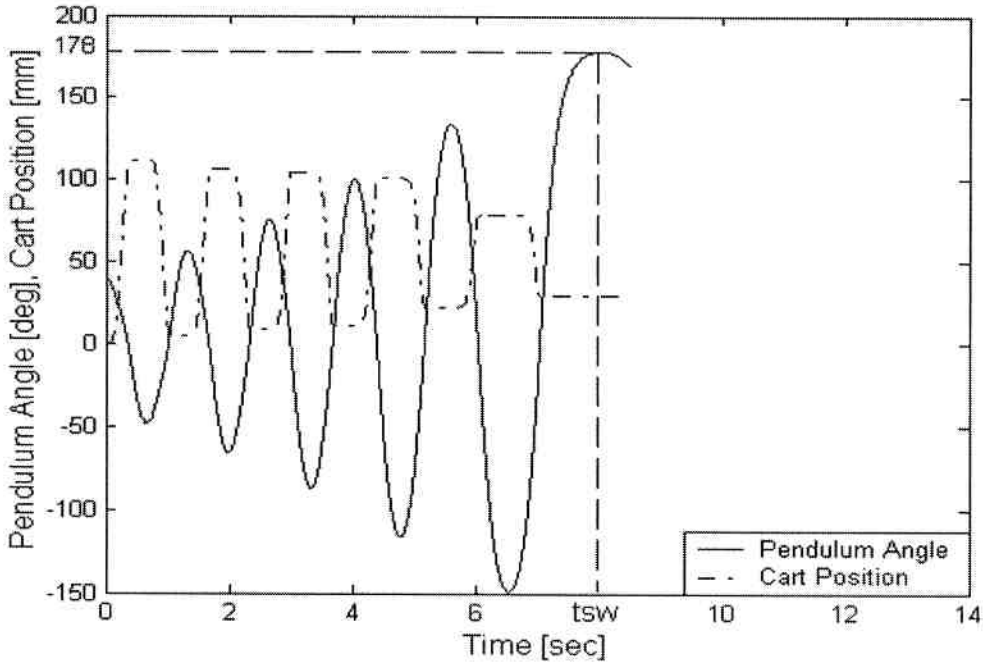
1 Pendulum swings downwards - cart moves in positive direction over prescribed trajectory
2. Pendulum swings upwards - cart is stationary



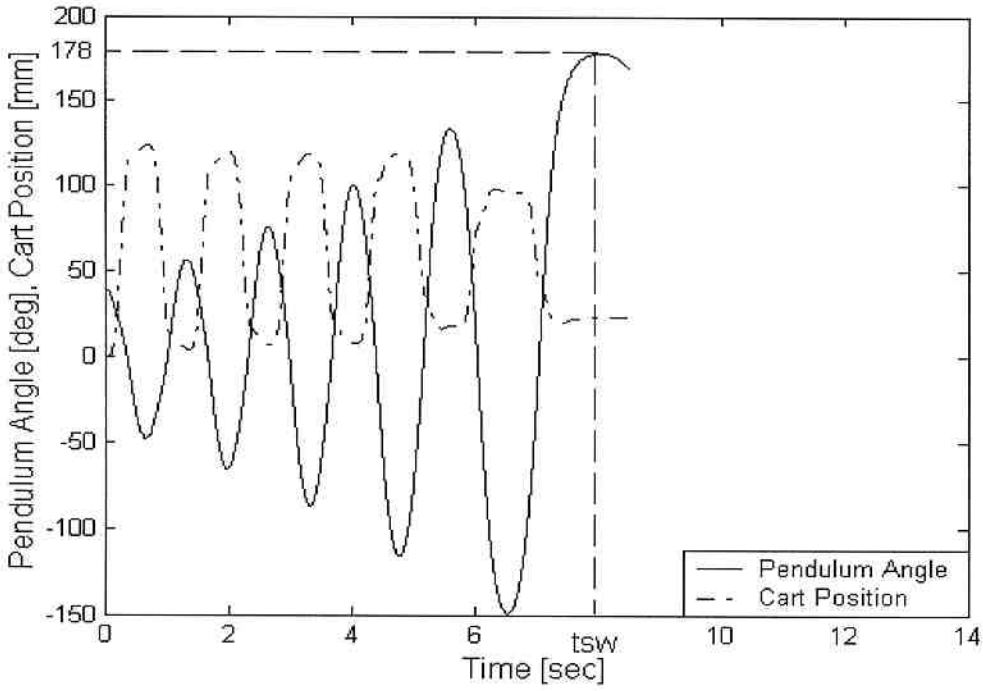
1 Pendulum starts swinging downwards. Cart is stationary until θtr is reached
2. Cart moves in negative direction over prescribed trajectory
3. Pendulum swings upwards - cart is stationary

Figure 5.11 Cart Motion Scenarios Depending on the Pendulum Swing Amplitude

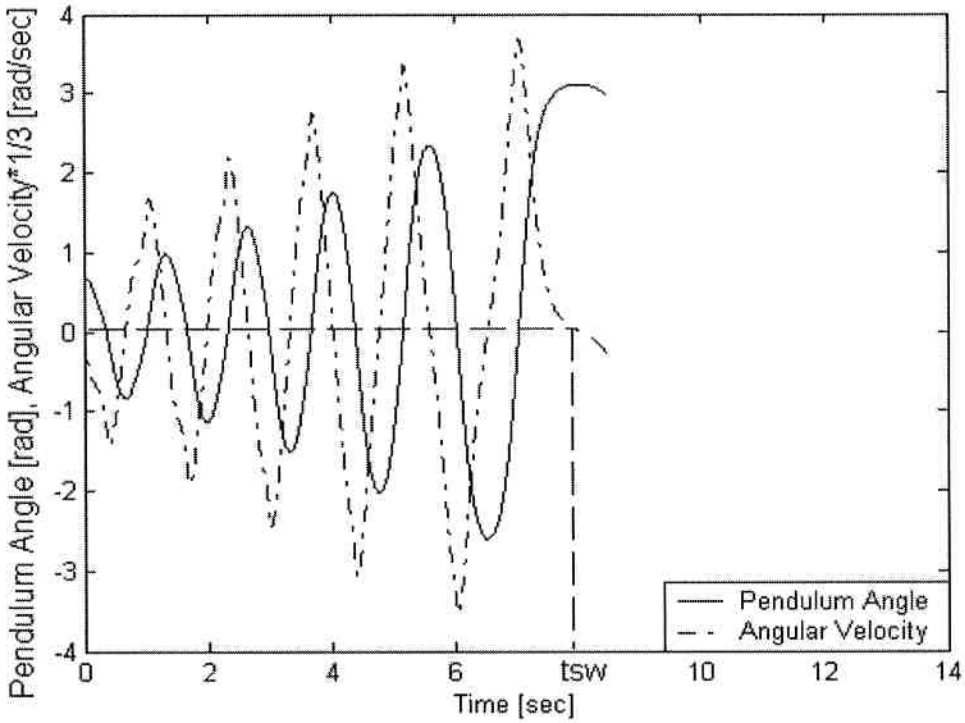
Graphs 5.11 – 5.14 present the results from the performed simulations.



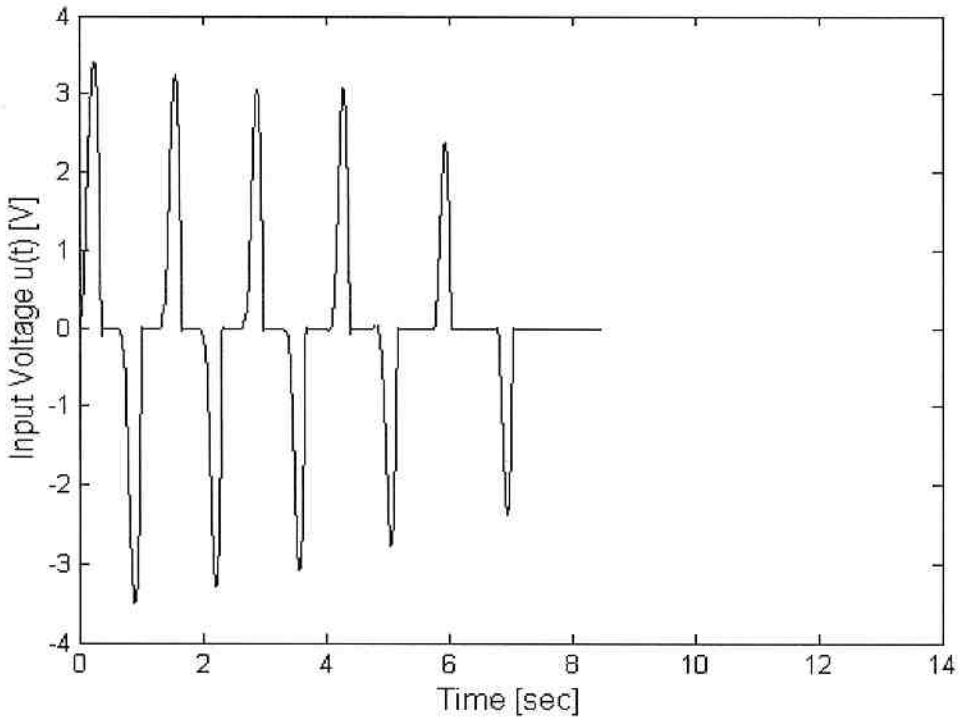
Graph 5.11 Cart Position (idealized) and Pendulum Angle Response under Energy-Based Control (simulations)



Graph 5.12 Cart Position and Pendulum Angle response under Energy- Based Controller (simulations)



Graph 5.13 Pendulum Angle vs. Pendulum Angular Velocity (Energy-Based Controller Simulations)



Graph 5.14 Control Signal (Energy-Based Controller Simulations)

Both Graphs 5.11 and 5.12 present the actual simulated response of the pendulum angle and cart position. The cart position in Graph 5.11 is idealized in a sense that the cart is motionless when the pendulum swings upwards and downwards with θ above θ_{tr}. This is done in order for a better impression of the controller to be established. Graph 5.12 depicts the actual (affected by the pendulum swinging) motion of the cart derived from the simulations.

The time when the control switches between the swing-up and stabilizing controller is marked in the graphs as t_{sw}. The pendulum angular velocity obtained from Graph 5.13 at $t=t_{sw}$ is less than 0.1 rad/s. This shows the satisfactory performance of the swing-up controller in the attempt to swing the pendulum gradually and to assure low pendulum

angular velocity at the moment of switching (Chapter 6 describes the stabilizing controller and presents a combined pendulum swing-up and stabilization simulation).

5.8 Experimental results

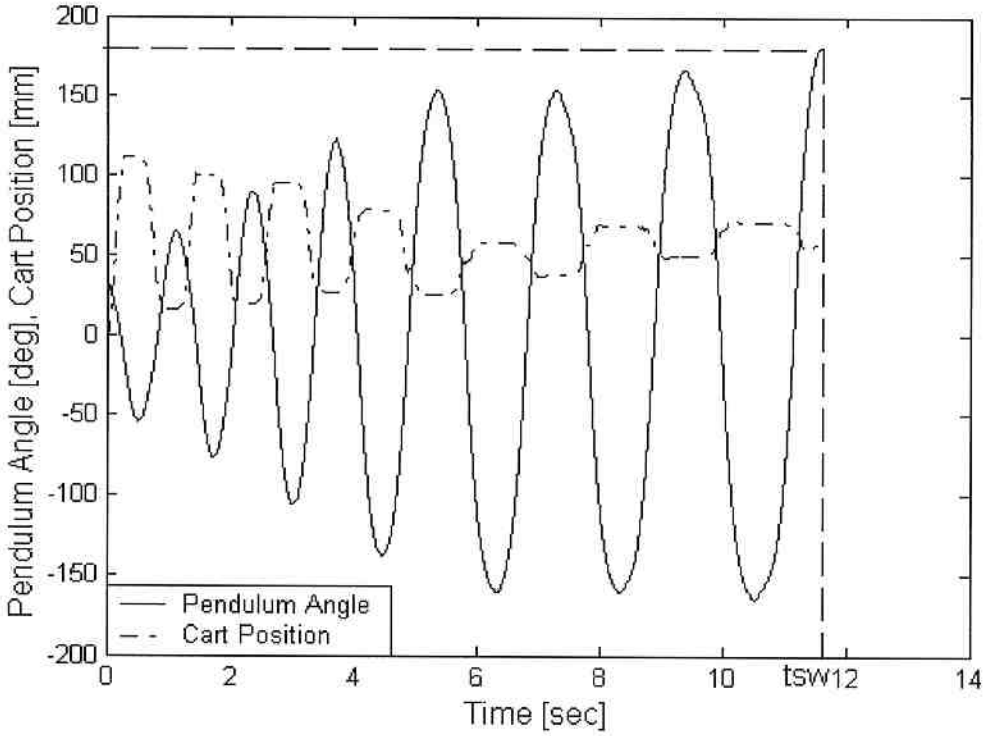
This subsection describes the experimental setup and discusses the system behavior when the EAC swing-up controller was applied to the actual physical apparatus.

During the experiments, a slightly different lookup table for entries of $Amp=Amp(\theta_{max})$ has been derived:

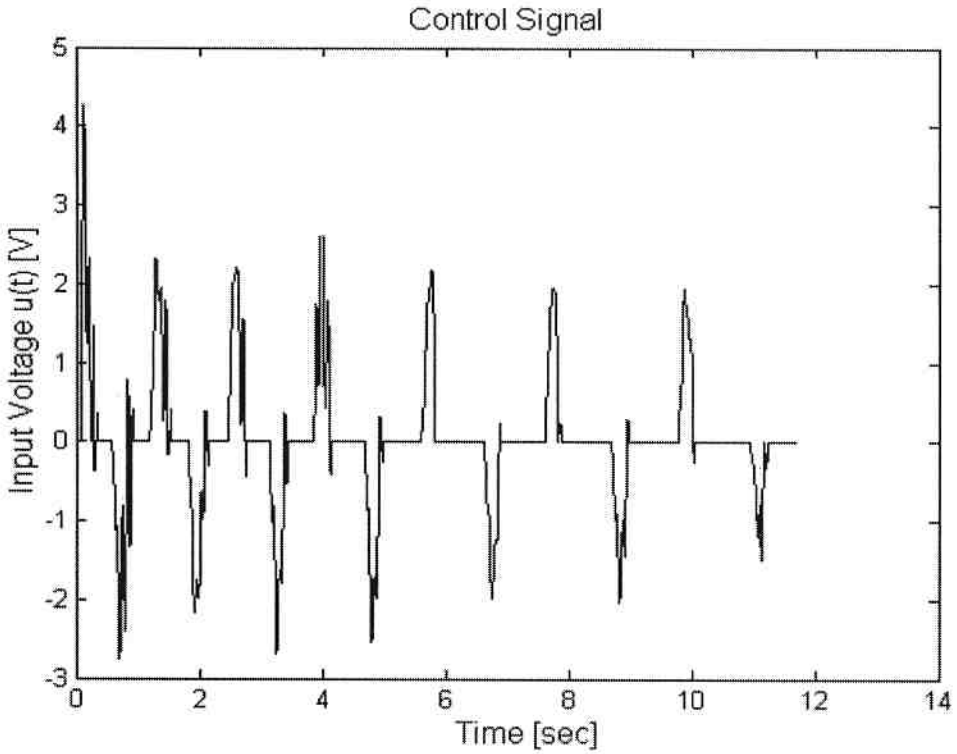
abs(θ_{max}) (deg)	0	60	110	130	150	165	180
Amp (mm)	60	40	30	25	20	17	10

Table 5.1 Lookup table defining values for cart excursion amplitude as a function of θ_{max} (experiments)

Graphs 5.15 and 5.16 are plots of the data gathered during real-time experiments.



Graph 5.15 Experimental System Performance



Graph 5.16 Control Signal (Experimental Results)

The experiment results follow similar to the simulations pattern until the pendulum starts swinging above ±145°. Then a few additional swings are performed in the range $150 < abs(\theta_{\max}) < 180$ to reach the inverted position smoothly. It should be noted that the unsymmetrical behavior of the cart-pole system noticeable in Graph 5.15 is mainly due to the some damages to the apparatus (including rail roughness), which in addition explains some of the spikes in the control signal (Graph 5.16).

(No data during the stabilization of the pendulum have been collected due to technical problems experienced at the moment of writing of the thesis)

The experiments with the physical pendulum apparatus showed the quite robust behavior of the controller in a sense that it ensures predictive and smooth swinging of the pendulum.

5.9 Chapter Concluding Remarks

The strategy for swinging up of the pendulum to its inverted position presented in this chapter is a result of a research motivated by the desire for the controller to be implemented on the physical apparatus.

The approach successfully reduced the nonlinear pendulum swing-up problem to a linear cart position controller design.

Defining the motion of the cart just when the pendulum swings downwards allows a smooth pendulum angle driven cart trajectory which provides the necessary acceleration/deceleration of the cart to be prescribed. Although that this method requires more pendulum swings compared to the fuzzy controller, the pendulum approaches the inverted position gradually and with low angular velocity, allowing graceful switching to the stabilization controller.

Careful process discrete-time simulations have been performed as a *proof of concept*.

The cart position controller has been designed to meet requirements set in the time domain by utilizing the root-locus method. However, significant care was taken to make sure that the entire system would have a suitable frequency domain behavior. The controller was implemented as a lead compensator. Lead compensation is commonly used in application where fast response is required. It provides small settling time and overshot. Another reason for using a lead compensator is that it yields a higher crossover frequency and provides additional phase lead, thereby improving the system's stability.

It should be noted that other controller implementations that are different than lead compensator can be implemented, but the lead compensator demonstrated a somehow optimal performance (considering the imposed time domain requirements to be met) during the simulations. For example a PID controller design was attempted in an effort to satisfy the requirements (5.27) and (5.28) but the model had too large of a system gain to be used with the current actuator.

Chapter 6

Stabilizing Linear State-Space Controller Design and Implementation

6.1 Chapter Introduction

This chapter addresses the implementation of a linear state-space controller, which assures stabilization of the pendulum in its inverted position.

The state-space design method is considered to be the most robust approach for controlling linear systems. [41, 43, 15, 16]. Furthermore, the procedure outlined by the state-space method for handling unstable systems is systematic and easily understood.

As the experimental results often differ from the theoretical formulation presented in the state-space theory, questions regarding suitable and optimal (theoretically derived) pole placement, as well as the velocity estimation task will be discussed.

Following the standard approach of state-space controller realization, the model of the system will be presented in state-space form linearized about the operating or regulating states. After considering controllability and observability of the system, a full-state feedback controller will be realized. State velocity estimation is recovered via state estimation.

Stabilizing linear state-space controller was first implemented on the cart/pole system at the University of Victoria by Ryan Mowat [39]. The following presentation includes further developments introduced to the controller both in terms of theoretical approach and implementation.

6.2 State Space Modeling for the Pendulum-Cart System

The standard linear state-space representation for the equation of motion of the cart-pendulum system in the inverted position of the pendulum is in the form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}u \end{aligned} \tag{6.1}$$

where with reference to Figure 6.1 the state vector is chosen as

$$\mathbf{x} = \begin{bmatrix} x_M \\ \theta \\ \dot{x}_M \\ \dot{\theta} \end{bmatrix} \tag{6.2}$$

The output vector $\mathbf{y}$ consists of the two directly measured states cart position x_M and pendulum angle θ:

$$\mathbf{y} = \begin{bmatrix} x_M \\ \theta \end{bmatrix} \tag{6.3}$$

The observed output vector $\mathbf{y}$ is independent of the control input u and thus from (6.1) and (6.3):

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.4}$$

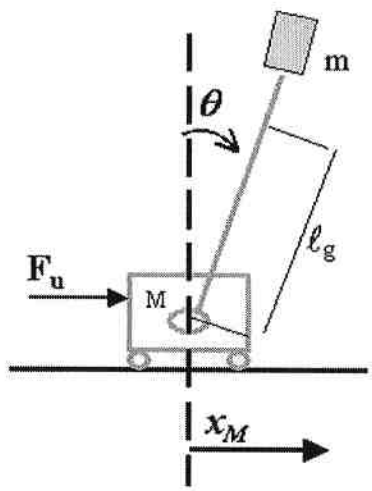


Figure 6.1 Pendulum on Cart system for Linear State-Space Control Modeling

6.2.1 Linearizations

In order to derive the constant and time invariant matrices **A** and **B,** the system model must be linearized about the equilibrium point ($\mathbf{x}_0$=0, u_0=0).

The system dynamic equations of motion as derived in Appendix B are:

$$\begin{cases} (M+m)\ddot{x}_M + ml_g \cos\theta\ddot{\theta} - ml_g\dot{\theta}^2 \sin\theta + b_c\dot{x}_M = \dfrac{A_G T_t}{r}u \\ \left(ml_g^2 + J_p\right)\ddot{\theta} + ml_g\ddot{x}_M \cos\theta - mgl_g \sin\theta + b_p\dot{\theta} = 0 \end{cases} \tag{6.5}$$

By solving (6.5) with respect to $\ddot{x}_M$ and $\ddot{\theta}$:

$$\ddot{x}_M = \frac{J_0(ml_g\dot{\theta}^2 \sin\theta - b_c\dot{x}_M) - ml_g \cos\theta(mgl_g \sin\theta - b_p\dot{\theta})}{M_0 J_0 - m^2 l^2 \cos^2\theta} +$$

$$+\frac{A_g T_t / rJ_0}{M_0 J_0 - m^2 l^2 \cos^2\theta} \equiv f_3(\theta, \dot{x}_M, \dot{\theta}) + g_3(\theta)u \tag{6.6}$$

$$\ddot{\theta} = \frac{1}{J_0}\left[-ml_g \cos\theta f_3(\theta, \dot{x}_M, \dot{\theta}) + mgl_g \sin\theta - b_p\dot{\theta}\right] \equiv$$

$$\equiv f_4(\theta, \dot{x}_M, \dot{\theta}) + g_4(\theta)u, \tag{6.7}$$

where $M_0 = M + m$ and $J_0 = J_p + ml_g^2$

Then the state-space representation of the system (with the state variables given in the system state vector (6.2)) is:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ f_3(x_2, x_3, x_4) + g_3(x_2)u \\ f_4(x_2, x_3, x_4) + g_4(x_2)u \end{bmatrix} \tag{6.8}$$

or, in vector form

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, u) \tag{6.9}$$

Expanding **F** by means of a Taylor series around an operating point ($\mathbf{x}_0$, u_0). yields

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}_0, u_0) + \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}_0 \\ u=u_0}} (\mathbf{x} - \mathbf{x}_0) + \left.\frac{\partial \mathbf{F}}{\partial u}\right|_{\substack{\mathbf{x}=\mathbf{x}_0 \\ u=u_0}} (u - u_0) + higher_order_terms \tag{6.10}$$

By noting that at the equilibrium point $\mathbf{F}(\mathbf{x}_0, u_0) = \mathbf{0}$ and neglecting the higher order terms in (6.10) the linearized model of the system about the equilibrium is obtained as:

$$\dot{\mathbf{x}} = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{0} \\ u=0}} \mathbf{x} + \left.\frac{\partial \mathbf{F}}{\partial u}\right|_{\substack{\mathbf{x}=\mathbf{0} \\ u=0}} u \tag{6.11}$$

Comparing (6.11) with (6.1) yields:

$$\mathbf{A} = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{0} \\ u=0}} \quad \text{and} \quad \mathbf{B} = \left.\frac{\partial \mathbf{F}}{\partial u}\right|_{\substack{\mathbf{x}=\mathbf{0} \\ u=0}} \tag{6.12}$$

By substituting the cart-pendulum system parameters in (6.13) by their physical values as defined in Table C1, Appendix C and carrying the calculations in the constant time-invariant matrices A and B are found as:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -1.8552 & -2.7889 & 0.0035 \\ 0 & 35.1557 & 8.4050 & -0.0664 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 6.3912 \\ -19.2614 \end{bmatrix} \tag{6.13}$$

With the matrices given by (6.13) the derivation of the linear state-space model of the cart-pendulum system in the inverted pendulum position is completed.

6.2.2. System Transfer Function

Taking the Laplace transform of (6.1) with the initial state $\mathbf{x}(0)$ equal to zero leads to:

$$\begin{aligned} \mathbf{X}(s) &= (s\mathbf{I} - \mathbf{A})\mathbf{B}u(s) \\ \mathbf{Y}(s) &= \mathbf{C}\mathbf{X}(s) \end{aligned} \tag{6.14}$$

then

$$\mathbf{Y}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}u(s) \tag{6.15}$$

and the transfer (matrix) function is:

$$\mathbf{H}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \tag{6.16}$$

Thus the characteristic polynomial of the system defined as det(s**I**-**A**) is:

$$s^4 + 2.86s^3 \ -35s^2 \ -82.45s \tag{6.17}$$

The zeroes of the characteristic polynomial, which represent the *open-loop* eigenvalues of the system are:

$$[-6.33 \quad -2.27 \quad 0 \quad 5.74] \tag{6.18}$$

Notice that the positive open-loop pole indicates open-loop instability. This unstable state is associated with the inverted position of the pendulum.

The transfer functions relating the cart position and the pendulum angle to the control input of the system can be obtained directly from (6.16):

$$\frac{x_M(s)}{u(s)} = \frac{6.39s^2 + 0.3567s - 188.9544}{s^4 + 2.86s^3 - 35s^2 - 82.45s}$$

$$\frac{\theta(s)}{u(s)} = \frac{-19.26s^2}{s^4 + 2.86s^3 - 35s^2 - 82.45s} \tag{6.19}$$

6.3 Controller Design

The *linear state feedback* for the single input inverted pendulum on a cart system is defined by the control law

$$u = -\mathbf{Gx} \tag{6.20}$$

and the closed-loop system is illustrated by Figure 6.2

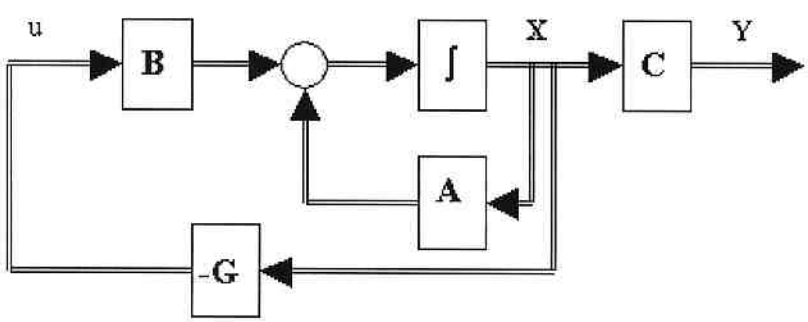


Figure 6.2 Linear State-Space System Closed-Loop Block Diagram

6.3.1 Controllability

The method of linear state feedback control is feasible if the open-loop dynamic system is controllable. Controllability is guaranteed theoretically by the algebraic theorem as outlined in [16]. Thus the time-invariant inverted pendulum on a cart system as given in Section 6.2 is controllable if and only if the rank r(**Q**) is equal to the order of the system. The controllability matrix **Q** is defined as:

$$\mathbf{Q} = [\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ \mathbf{A}^3\mathbf{B}] = \begin{bmatrix} 0 & 6.40 & -17.9 & 85.8 \\ 0 & -19.3 & 55.0 & -831.2 \\ 6.40 & -17.9 & 85.8 & -344.3 \\ -19.30 & 55.0 & -831.2 & 2710.0 \end{bmatrix} \tag{6.21}$$

The number of linearly independent columns in the matrix **Q** is four and thus the rank of **Q**, r(**Q**), is four, which is equal to the order of the system. Subsequently, the system is controllable.

6.3.2 Pole Placement for the Closed – Loop Inverted Pendulum System

The closed-loop dynamic system is determined by substituting (6.20) in (6.1):

$$\begin{aligned} \dot{\mathbf{x}} &= (\mathbf{A} - \mathbf{BG})\mathbf{x} \\ \mathbf{y} &= \mathbf{Cx} \end{aligned} \tag{6.22}$$

Consequently, the closed-loop poles of the system are the eignevalues of the matrix **A-BG,** defined by

$$|s\mathbf{I} - \mathbf{A} + \mathbf{BG}| = 0 \tag{6.23}$$

The pole placement task is to determine the form of the matrix **G**, $G \in M_{1x4}(R)$ which will result in a closed-loop system with poles in the desired location.

The determination of the matrix **G** can be approached by either arbitrary or optimal pole-placement methods. Arbitrary pole placement methods include Ackerman [41], Bass-Gurra's [16].

An optimal pole-placement algorithm that is often considered is the Linear Quadratic Regulator design [14,16]. The LQR method determines the optimal control solution for the feedback state-space system (6.22) by finding the control gain matrix **G**, which minimizes the steady state performance (cost) function:

$$V = \int_t^{\infty} (\mathbf{x'Qx} + u'\mathbf{R}u)d\tau \tag{6.24}$$

where **Q** and **R** are symmetric matrices.

The form for the cost function (6.24) can be expressed as:

$$V = \mathbf{x'Mx} \tag{6.25}$$

where **M** satisfies the algebraic *Ricatti* equation:

$$\mathbf{0} = \mathbf{MA} + \mathbf{A'M} - \mathbf{MBR^{-1}B'M} + \mathbf{Q}$$

and the optimum gain is given by

$$\mathbf{G} = \mathbf{R}^{-1}\mathbf{B}'\mathbf{M} \tag{6.26}$$

The selection of the matrices **Q** and **R** is designer specific. In the case of the inverted pendulum with state vector $\mathbf{x} = [x_M \quad \theta \quad \dot{x} \quad \dot{\theta}]^T$ if the penalty on the deviation of the cart's position x_M from the origin is high, then the diagonal element of **Q** associated with state x_M should be high. Likewise the relative cost associated with controller actuation can be specified by appropriate choice of **R**.

If the **Q** and **R** matrices are defined as:

$$\mathbf{Q} = \begin{bmatrix} 3e6 & 0 & 0 & 0 \\ 0 & 150 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1200 \end{bmatrix} \tag{6.27}$$

and

$$\mathbf{R} = [2000],$$

the LQR method applied on the inverted pendulum yields a suitable gain matrix G

$$\mathbf{G} = -[38.73 \quad 43.68 \quad 18.62 \quad 7.98] \tag{6.28}$$

which leads to a closed-loop system with eigenvalues (found from Equation (6.23)):

$$eigenvalues_{close_loop} = [-12.74 + j7.03,\ \ -12.74\ -j7.03,\ -7.20,\ \ -4.79\ ] \quad (6.29)$$

The closed-loop eignevalues determined by the LQR method can be used as a base for a direct numerical pole-placement algorithm for experimental optimization of the controller's performance. That is, the closed loop system's dynamics can be optimized by direct adjustment of the eigenvalues (6.29). Experimental optimization of the pole locations is illustrated below in Section 6.6.

6.4. Digital System Model

In order a real-time digital controller to be designed the continuous model of the system presented by equations (6.1) to (6.4) and the controller (6.20) must be transformed to their discrete equivalent.

When the zero-order hold transformation [14] with period of discretization T is applied, the difference equations for the discrete system are:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Phi}(k) + \mathbf{\Gamma}u(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned} \quad (6.30)$$

where

$$\mathbf{\Phi} = e^{\mathbf{A}T}$$

$$\mathbf{\Gamma} = \int_0^T e^{\mathbf{A}t}\mathbf{B}dt \tag{6.31}$$

The control law at given time k is given by:

$$u(k) = -\mathbf{K}\mathbf{x}(\mathrm{k}) \tag{6.32}$$

Which results in the closed-loop system:

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) - \mathbf{\Gamma}\mathbf{K}\mathbf{x}(k) = \left[\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K}\right]\mathbf{x}(k) \tag{6.33}$$

The z transform of (6.33) is :

$$(z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K})\mathbf{X}(z) = 0 \tag{6.34}$$

and the closed-loop characteristic equation is:

$$\left|z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K}\right| = 0 \tag{6.35}$$

When the desired continuous closed-loop pole locations are determined (in the form (6.29)), the equivalent digital pole locations in the z plane are found from the transformation [14]

$$poles_{digital} = e^{T * poles_{cont}} \tag{6.36}$$

The discrete poles that are equivalent to the continuous poles determined by the LQR method for optimal stabilization control of the inverted pendulum as given by (6.28) are:

$$poles_{digital} = [0.9747 + j0.0137,\ \ 0.9747 - j0.0137,\ 0.9857,\ \ 0.9905] \tag{6.37}$$

The control-law design, then, consists of finding the elements of **K** such that the roots of (6.35), that is, the poles of the closed-loop system, are in the desired locations (6.37). This can be done, by utilizing a pole placement algorithm, such as the Ackerman's formula, (or by using some of the methods implemented in Matlab). The control gain matrix **K** for the digital controller is found to be:

$$\mathbf{K} = -[37.4129\ \ 42.4379\ \ 18.0409\ \ \ 7.7479] \tag{6.38}$$

6.5 State Estimation Issues

The controller design presented in the previous section is based on the assumption that the whole state vector **x** is available. It was shown in Chapter 2 that the cart-pendulum

apparatus is equipped with two sensors which measure the cart's position and the pendulum's angle. Thus the two states $[x_M \ \theta]$ in the system model (6.1) are directly available. State estimation is necessary in order the cart's velocity and pendulum's angular velocity (states $[\dot{x} \ \dot{\theta}]$) to be determined. The focus of this section is to provide theoretical background for velocity estimation. All of the presented derivations are directly for the discrete time case. Presented in order methods for velocity estimation, with discussions for their advantages and disadvantages, are the *Prediction and* the *Current Full Order Estimators* and the *Predictor Reduced Order Estimator* in an attempt to form a roadmap for the definition of the so called *Integral Observer* which is an *reduced order current* velocity estimator.

6.5.1 System Observability

In order for the model based velocity estimation methods to be applied the system model must meet the observability criteria.

As outlined in [16] a system is theoretically observable if and only if the algebraic observability criteria are satisfied, i.e. the rank of the observability matrix, r(**N**), must equal the order of the system. The observability matrix for the 4th order cart-pendulum system is defined as:

$$\mathbf{N} = \begin{bmatrix} \mathbf{C}^T & \mathbf{A}^T\mathbf{C}^T & \left(\mathbf{A}^T\right)^2\mathbf{C}^T & \left(\mathbf{A}^T\right)^3\mathbf{C}^T \end{bmatrix} \tag{6.39}$$

Matrices **A** and **C** are determined by the model of the system given in Section 6.2 and defined by (6.14) and (6.4) respectively. With matrix **A** and **C** such defined the controllability matrix is:

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1.8552 & 35.1557 & 5.2970 & -17.9259 \\ 0 & 0 & 1 & 0 & -2.7889 & 8.4050 & 7.8074 & -23.9984 \\ 0 & 0 & 0 & 1 & 0.0035 & -0.0664 & -1.8652 & 35.1895 \end{bmatrix} \quad (6.40)$$

and has rank, r(**N**)=4, and thus the cart-pendulum system is observable.

6.5.3 Full Order Observers

6.5.3.1 Prediction estimator

One method of estimating the state vector is to construct a model of the dynamic plant system:

$$\overline{\mathbf{x}}(k+1) = \mathbf{\Phi}\overline{\mathbf{x}}(k) + \mathbf{\Gamma}u(k) \quad (6.42)$$

As Φ, Γ and *u(k)* are known this *open-loop* estimator should work if the correct state $\mathbf{x}(0)$ can be obtained and $\overline{\mathbf{x}}(0)$ is set to be equal to it. The basic problem here is that the estimator is running open-loop and not utilizing any continuing measurements of the

system's behavior and it may be expected to diverge from the correct value. A direct solution to this problem is to feed back the difference between the measured output and the estimate output and to constantly correct the model with this error signal:

$$\bar{\mathbf{x}}(k+1) = \mathbf{\Phi}\bar{\mathbf{x}}(k) + \mathbf{\Gamma}u(k) + \mathbf{L}_p\left[\mathbf{y}(k) - \mathbf{C}\bar{\mathbf{x}}(k)\right] \tag{6.43}$$

Equation (6.43) is direct application of the analog full-order estimator for the digital control task. The estimator thus determined is a *prediction estimator* because measurement at time k results in an estimate that is valid at time $k+1$.

A difference equation describing the behavior of the system error, $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$, is obtained by subtracting (6.43) from (6.30):

$$\tilde{\mathbf{x}}(k+1) = \left[\mathbf{\Phi} - \mathbf{L}_p\mathbf{C}\right]\tilde{\mathbf{x}}(k) \tag{6.44}$$

which has the characteristic equation:

$$\left|z\mathbf{I} - \mathbf{\Phi} + \mathbf{L}_p\mathbf{C}\right| = 0 \tag{6.45}$$

If the matrix $\left[\mathbf{\Phi} - \mathbf{L}_p\mathbf{C}\right]$ is a stable system matrix, $\bar{\mathbf{x}}(k)$ will converge to $\mathbf{x}(k)$ regardless of the value of $\bar{\mathbf{x}}(0)$ and the speed of convergence depends on the values of the eigenvalues of the system matrix.

After suitable eigenvalues for the system matrix $\left[\mathbf{\Phi} - \mathbf{L}_p\mathbf{C}\right]$ (defined as roots of the characteristic equation (6.45)) are selected, with a minor modification, the estimator gain

matrix **Lp** can be found by using the same approach as described for finding the controller gain **K** in Section 6.4. The modification appears as taking the transponse of the system matrix:

$$\left[\mathbf{\Phi} - \mathbf{L}_p\mathbf{C}\right]^T = \mathbf{\Phi}^T - \mathbf{C}^T\mathbf{L}_p^T \tag{6.46}$$

The result in (6.45) is in the same form as the system matrix $\mathbf{\Phi} - \mathbf{\Gamma K}$ of the control problem in section 6.4. Therefore, to find $\mathbf{L}_p^T$ (and consecutively $\mathbf{L}_p$), $\mathbf{\Phi}^T, \mathbf{C}^T$ and $\mathbf{L}_p^T$ are substituted for $\mathbf{\Phi}, \mathbf{\Gamma}$ and $\mathbf{K}$ and the control-design result is used.

6.5.3.2. Full Order Current Estimators

As derived in the previous subsection, the predictor estimator arrives at the current estimate $\bar{\mathbf{x}}(k)$ after receiving measurements up through $y(k\text{-}1)$. This means that the current value of the control does not depend on the most recent value of the observation and thus it might not be as accurate as it could be. For controllers implemented on slow computers this delay between the observation sample and the validity time for the control signal can be advantageous in sense that it will provide the time necessary time to compute the control value. On the other hand, in many systems having high computational power, the computational time is quite short when compared to the sample period, and the delay of almost a cycle between measurement and the proper time for applying the control is an unnecessary waste and may lead to system instability. To

overcome this, an estimator that provides a current estimate $\hat{\mathbf{x}}$ can be construct by modifying Equation (6.43):

$$\hat{\mathbf{x}}(k) = \overline{\mathbf{x}}(k) + \mathbf{L}_c\left[\mathbf{y}(k) - \mathbf{C}\overline{\mathbf{x}}(k)\right] \tag{6.47}$$

where $\overline{\mathbf{x}}(k)$ is the predicted estimate based on a model prediction from the previous time estimate:

$$\overline{\mathbf{x}}(k) = \mathbf{\Phi}\overline{\mathbf{x}}(k-1) + \mathbf{\Gamma}u(k-1) \tag{6.48}$$

In practice, this estimator is impossible to implement as some finite time is required to perform the necessary calculations for the state estimations and to calculate the control output. However, with appropriate arrangement of the program code and with software running on a fast computer, computation delays can be minimized. Again, as with the predictor estimator, the estimator gain matrix $\mathbf{L}_c$ is computed from the difference equation describing the dynamics of the system error. The difference equation is found by subtracting (6.47) from (6.30):

$$\tilde{\mathbf{x}}(k+1) = \left[\mathbf{\Phi} - \mathbf{L}_c\mathbf{H}\mathbf{\Phi}\right]\tilde{\mathbf{x}}(k) \tag{6.49}$$

After suitable estimator eigenvalues for the system matrix $\left[\mathbf{\Phi} - \mathbf{L}_c\mathbf{H}\mathbf{\Phi}\right]$ are selected, $\mathbf{L}_c$ can be determined by an appropriate pole-placement procedure.

6.5.4 Reduced – Order Predictor Estimators

In contrast to the full-order estimator, the states that are directly measured are not re-estimated by the reduced-order estimator. A direct benefit of the reduced-order estimator is that it is computationally less expensive than the full-order estimators. The reduced order estimator is a prediction estimator and does not have a current estimator counterpart.

The form of the reduced-order estimator can be derived by following the approach derived by [14]. Thus, the digital state-space model in (6.29) must be first partitioned by collecting the measured states, denoted by subscript a, and the unmeasured states denoted by subscript b:

$$\begin{bmatrix} \mathbf{x}_a(k+1) \\ \mathbf{x}_b(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_{aa} & \mathbf{\Phi}_{ab} \\ \mathbf{\Phi}_{ba} & \mathbf{\Phi}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{x}_a(k) \\ \mathbf{x}_b(k) \end{bmatrix} + \begin{bmatrix} \mathbf{\Gamma}_a \\ \mathbf{\Gamma}_b \end{bmatrix} u(k) \tag{6.50}$$

The reduced order predictor estimator is given as:

$$\hat{\mathbf{x}}_b(k+1) = \mathbf{\Phi}_{bb}\hat{\mathbf{x}}_b(k) + \mathbf{\Phi}_{ba}\mathbf{x}_a(k) + \mathbf{\Gamma}_b u(k) + \mathbf{L}_r\left[\mathbf{x}_a(k+1) - \mathbf{\Phi}_{aa}\mathbf{x}_a(k) - \mathbf{\Gamma}_a u(k) - \mathbf{\Phi}_{ab}\hat{\mathbf{x}}_b(k)\right]$$

with the estimator error dynamics determined by the difference equation:

$$\tilde{\mathbf{x}}_b(k+1) = \left[\mathbf{\Phi}_{bb} - \mathbf{L}_r \mathbf{\Phi}_{ab}\right]\tilde{\mathbf{x}}(k)_b \tag{6.51}$$

Determination of the reduced-order estimator gain matrix **L**r is done exactly as before – by picking suitable eigenvalues for the matrix $\left[\mathbf{\Phi}_{bb} - \mathbf{L}_r \mathbf{\Phi}_{ab}\right]$ and the by applying a pole-placement method.

6.5.5 Integral Observer

Since the general ideas for state estimation have been covered, this subsection discusses the implementation of an integral observer, which is the estimator used in the software to recover the unmeasured states (cart and pendulum velocity) for the inverted pendulum system. The theoretical presentation outlines the estimator design as described in the paper from Erlic and Lu [11]. The integral observer is of the same order as the reduced-order estimator but it is a current estimator and thus combines the advantages for both the full-order current estimator and the reduced-order prediction estimator as described above.

The continuous *4x4* inverted pendulum state-space model, as presented in (6.1), with 2 unmeasured velocities can be rewritten in terms of its second derivatives,

$$\begin{bmatrix} \ddot{x}_M \\ \ddot{\theta} \end{bmatrix} = \overline{\mathbf{F}} \begin{bmatrix} \dot{x}_M \\ \dot{\theta} \end{bmatrix} + \overline{\mathbf{G}} \begin{bmatrix} x_M \\ \theta \end{bmatrix} + \overline{\mathbf{H}} u, \tag{6.52}$$

where

Equation (6.52) can be presented in matrix form as:

$$\dot{\mathbf{z}} = \overline{\mathbf{F}}\mathbf{z} + \overline{\mathbf{G}}\mathbf{y} + \overline{\mathbf{H}}u \tag{6.53}$$

Vector **y** is the output vector as before and vector **z** is defined as state velocities:

$$\mathbf{y} = \begin{bmatrix} x_M \\ \theta \end{bmatrix} \quad ; \qquad \mathbf{z} = \begin{bmatrix} \dot{x}_M \\ \dot{\theta} \end{bmatrix} \tag{6.54}$$

Once again, the velocity estimation task is determining an estimate for the state velocities z defined as:

$$\hat{\mathbf{z}} = \begin{bmatrix} \dot{\hat{x}}_M \\ \dot{\hat{\theta}} \end{bmatrix} \tag{6.55}$$

and thus the estimation state equation can be chosen as:

$$\dot{\hat{\mathbf{z}}} = \overline{\mathbf{F}}\hat{\mathbf{z}} + \overline{\mathbf{G}}\mathbf{y} + \overline{\mathbf{H}}u + \mathbf{L}_i(\mathbf{z} - \hat{\mathbf{z}}) \tag{6.56}$$

where the estimator –error dynamics

$$\mathbf{z}_{error} = \mathbf{z} - \hat{\mathbf{z}} \tag{6.57}$$

are determined by subtracting (6.56) from (6.53). With a suitable choice of $\mathbf{L}_i$ it can be shown that the estimation error exponentially decays to zero:

$$\dot{\mathbf{z}}_{error} = \left(\overline{\mathbf{F}} - \mathbf{L}_i\right)\mathbf{z}_{error} \tag{6.58}$$

Digital implementation of the integral observer is done by integration of the state equation (6.55) for one sample period T

$$\int_{t-T}^{t} \dot{\hat{\mathbf{z}}}(\tau)d\tau = \int_{t-T}^{t} \left(\overline{\mathbf{F}}\hat{\mathbf{z}}(\tau) + \overline{\mathbf{G}}\mathbf{y}(\tau) + \overline{\mathbf{H}}u(\tau) + \mathbf{L}_i\left(\mathbf{z}(\tau) - \hat{\mathbf{z}}(\tau)\right)\right)d\tau \tag{6.59}$$

The integration of (6.56) as indicated in (6.59) is carried out term by term

$$\int_{t-T}^{t} \dot{\hat{\mathbf{z}}}(\tau)d\tau = \hat{\mathbf{z}}(kT) - \hat{\mathbf{z}}(kT-T)$$

$$\int_{kT-T}^{T} \overline{\mathbf{F}}\mathbf{z}(\tau)d\tau \cong T\overline{\mathbf{F}}\mathbf{z}(kT-T)$$

$$\int_{kT-T}^{T} \overline{\mathbf{G}}\mathbf{y}(\tau)d\tau \cong T\overline{\mathbf{G}}\mathbf{y}(kT-T)$$

$$\int_{kT-T}^{T} \overline{\mathbf{H}}u(\tau)d\tau \cong T\overline{\mathbf{H}}u(kT-T) \tag{6.60}$$

$$\mathbf{L}_i\left(\int_{kT-T}^{kT} \mathbf{z}(\tau)d\tau - \int_{kT-T}^{kT} \hat{\mathbf{z}}(\tau)d\tau\right) = \mathbf{L}_i\left(\mathbf{y}(kT) - \mathbf{y}(kT-T)\right) - \mathbf{L}_i \int_{kT-T}^{kT} \hat{\mathbf{z}}(\tau)d\tau$$

$$\cong \mathbf{L}_i\left(\mathbf{y}(kT) - \mathbf{y}(kT-T) - T\hat{\mathbf{z}}(kT-T)\right)$$

As in the previous sections, it will be referred to k and k-1 as the current and previous samples respectively, i.e. the sampling period T will be omitted for simplicity. After collecting terms, and solving for $\hat{\mathbf{z}}(k)$, the integral velocity estimator in finite difference form is:

$$\hat{\mathbf{z}}(k) \cong \overline{\mathbf{F}}_o \hat{\mathbf{z}}(k-1) + \overline{\mathbf{H}}_o u(k-1) + \overline{\mathbf{G}}_o \mathbf{y}(k) + \mathbf{L}_i\big(\mathbf{y}(k) - \mathbf{y}(k-1)\big) \quad (6.61)$$

where,

$$\begin{aligned} \overline{\mathbf{F}}_o &= \mathbf{I} + T(\overline{\mathbf{F}} - \mathbf{L}_i) \\ \overline{\mathbf{H}}_o &= T\overline{\mathbf{H}} \\ \overline{\mathbf{G}}_o &= T\mathbf{G} \end{aligned} \quad (6.62)$$

As shown in (6.61) the current time estimation of the unmeasured states includes current samples of the output and thus the integral state velocity estimator is a reduced order current estimator, rather than predictor estimator. The integral estimator is advantageous because the position signal errors are minimized by multiplication of the sampling period in the first term in (6.61). In the last term, the position errors are reduced by exact integration as derived in (6.60).

A similar design procedure based on optimal control, as done for the controller design, can be followed for deriving the form of the estimator gain matrix $\mathbf{L}_i$. And as before the eigenvalues of the resulting system may be varied in search of satisfactory performance. Best experimental results are achieved with real integral estimator eigenvalues having magnitudes that are approximately twice the magnitude of the controller eigenvalues:

$$\text{estimator_eigenvalues} = -\,[20\ 25] \quad (6.63)$$

In order for the estimator eigenenvalues to be as in (6.63) the matrix $\mathbf{L}_i$ must be:

$$\mathbf{L}_i = \begin{bmatrix} 17.2111 & 0.0035 \\ 8.4050 & 24.9336 \end{bmatrix} \tag{6.64}$$

6.6 Regulator Design

The controller developed in Section 6.4 and the integral observer from Section 6.5.6 are designed independently. That is the design of the controller is performed with the assumption that the true state $\mathbf{x}$ is fed back instead of the estimate. According to the separation principle as outlined in [14] this is a valid assumption since the poles of the complete system consisting of the estimator feeding the control law will be the same as for the two cases analyzed separately.

The overall discrete time block-diagram of the system is given in Figure 6.3

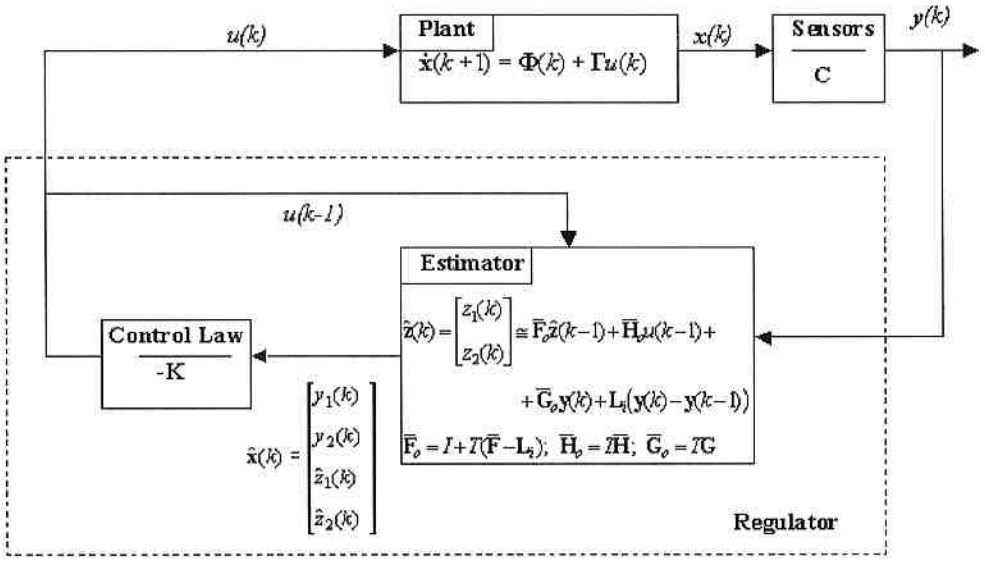


Figure 6.3 Discrete-Time Closed-Loop System Block Diagram

and Figure 6.4 presents the simulated behavior of the system. The simulations shown in Figure 6.5 are result of combining the swing-up controller presented in the previous Chapter 5 and the linear state-space stabilizing controller.

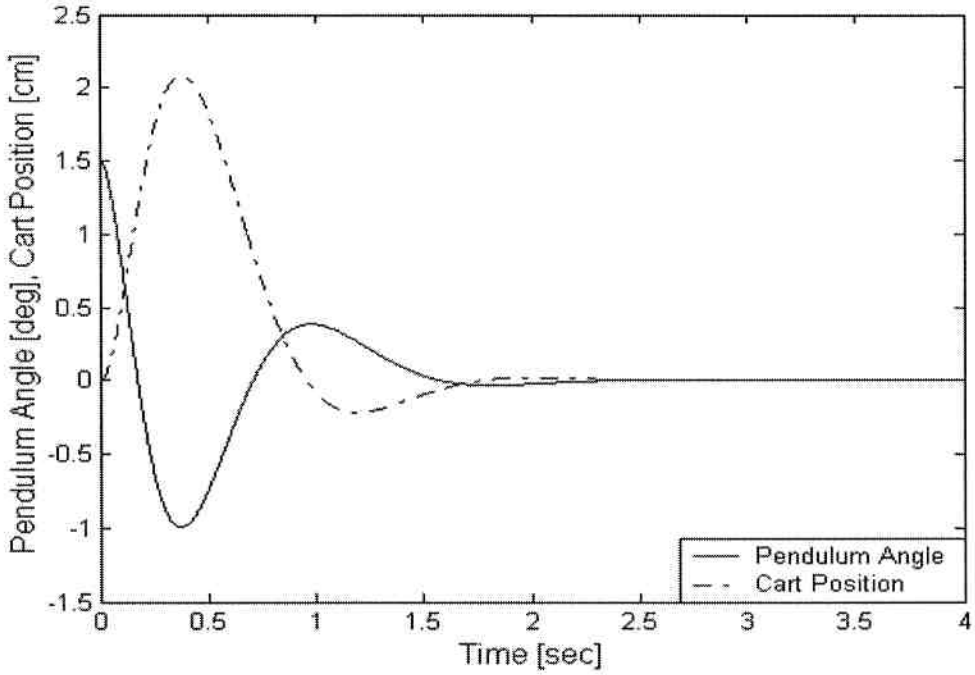


Figure 6.4 Simulated System Response to Stabilizing Controller with Integral Observer

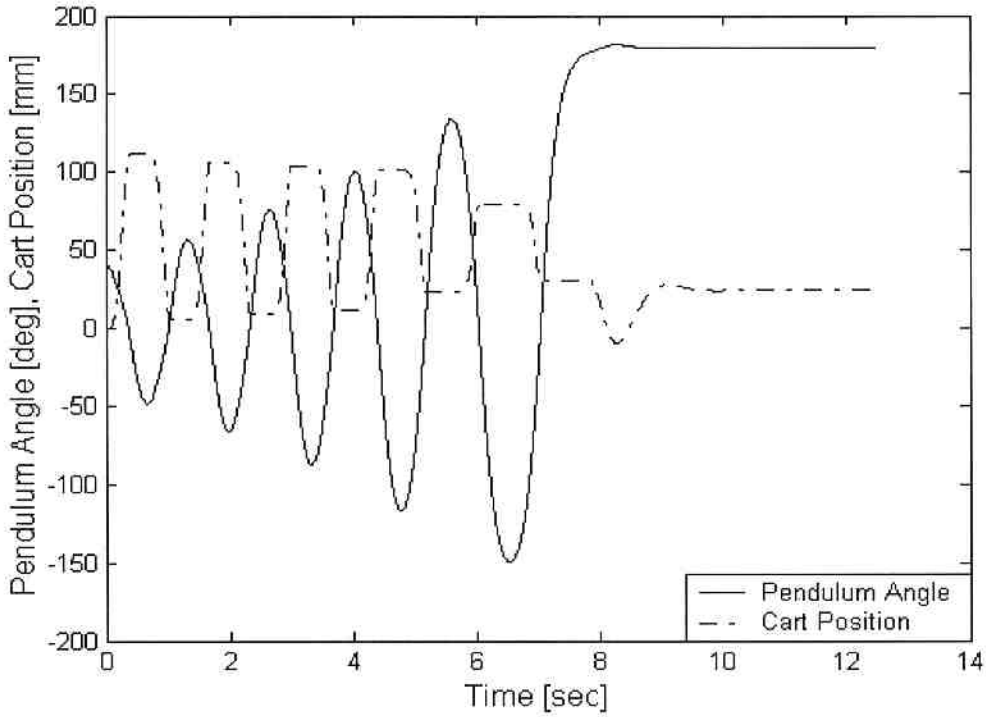


Figure 6.5 Overall Pendulum Swing-up and Stabilization Simulation

6.6.1 Experimental closed-loop pole adjustment

The locations of the closed-loop system poles determined by the LQR method in Section 6.3.2 can be further experimentally optimized and the associated continuous control gain matrix **G** found by a pole placement method.

Some discussions on the pole selection relating the suggestions for dynamic response shaping in the control texts and the experiments are given below [41, 16, 15].

- The feedback gains are proportional to the amount that the poles are to be moved. This can be seen directly from the Bass-Gura formula. Thus the less poles are moved the smaller the gain matrix is. Thus, large system gains are avoided by limiting the changes in the coefficients of the characteristic equation. It should also be noted that the feedback control gains are inversely proportional to the controllability of the system – the more unstable the system, the larger the gains that are needed. As a consequence, the best selection of closed-loops poles is obtained by achieving a stable system with the least amount of modification of the original systems dynamics. In the case of the inverted pendulum with open-loop eigenvalues [0 5.74 -6.33 -2.27], the dynamics associated with the eigenvalues [0 5.74] are not acceptable and must be altered. However the eigenvalues [-6.33 –2.82] may be acceptable. An arbitrary selection of poles in order of [-80 to –200] could require lead to actuator saturation and thus cause instability.
- Complex eigenvalues theoretically provide the opportunity to specify the desired damping of the system and thus to modify the transient response of the system. On the other hand the experimental results prove that better robustness to external

disturbance and minimization of the steady-state error are achieved with completely real eigenvalues.

After a sequence of experiments were held observing the considerations giving above and the pole locations prescribed by the LQR method, the continuous closed-loop poles and the corresponding controller gain matrix giving best system performance were found to be:

$$poles_{cont} = -\,[13.05 \;\; 12.05 \;\; 7.27 \;\; 6.04] \tag{6.65}$$

The equivalent digital system closed-loop poles and control gain matrix are:

$$poles_{digit} = [0.9742 \quad 0.9762 \quad 0.9856 \quad 0.9880] \tag{6.67}$$

$$\mathbf{K} = -[35.26 \;\; 40.8177 \;\; 16.85 \;\; 7.39] \tag{6.68}$$

During the experiments the gain matrix of the integral observer was kept unchanged as given by (6.64) and that is the actual form of the matrix implemented in the software.

6.7 Concluding Remarks Concerning Linear State-Space Control

The state-space method provides an excellent method for controller design for unstable system regulation. Stability, controllability and observability are all easily determined.

A stable set of linear feedback gains (**G**) can be determined for the inverted pendulum stabilizing task by using optimal LQR method. However, further experimental adjustments of the controller gains are required to achieve a high precision controller.

Regarding the state-space estimator design combining the advantages of both reduced-order and current estimators, the integral observer proved to be the best solution for the overall controller realization.

Chapter 7

Software Implementation of the Controller

7.1 Chapter Introduction

This chapter describes the development of the software application implementing the control algorithms and providing the user interface. Presented, also, are some of the key features of the QNX Neutrino - the underlying operating system. These features are utilized in the successful development of the pendulum controller application. The application itself is built using a *client / server* architecture. This is done mainly in an attempt to separate the functionality and thus to achieve modularity of the system.

The *Client* application provides the interface to the user. The server application implements the swing-up and stabilizing controllers as described in Chapters 5 and Chapter 6 respectively.

7.2 The Operating System Platform

QNX Neutrino 6.2.1 has been chosen as a platform for running the pendulum control application. Neutrino is a high performance hard real time operating system implementing the POSIX standard and thus becomes an ultimate choice for control applications since it provides a deterministic clock period and facilitates the application code development. Below are presented some of the key features of QNX Neutrino and features utilized in the development of the application program

Microkernel Architecture

Neutrino is highly modular OS that features a protected microkernel. A microkernel OS is structured as a tiny kernel that provides the minimal services used by a team of optional cooperating processes, which in turn provide the higher-level OS functionality. The microkernel itself lacks filesystems and many other services normally expected of an OS - those services are provided by optional processes. The real goal in designing a microkernel OS is not simply to "make it small." A microkernel OS embodies a fundamental change in the approach to delivering OS functionality. *Modularity is the key, size is but a side effect.* Since the IPC (Inter Process Communication) services provided by the microkernel are used to "glue" the OS itself together, the performance and flexibility of those services govern the performance of the resulting OS. With the exception of those IPC services, a microkernel is roughly comparable to a realtime executive, both in terms of the services provided and in their real-time performance.

As the following diagrams show, the microkernel offers *complete memory protection*, not only for user applications, but also for OS components (device drivers, filesystems, etc.)

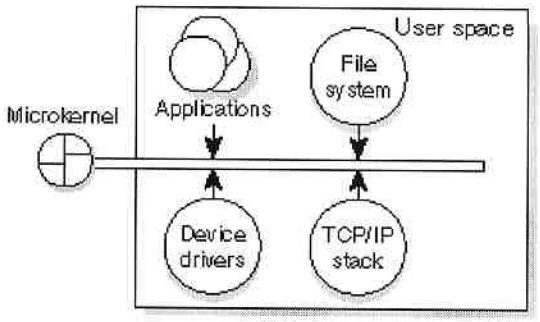


Figure 7.1 The QNX Microkernel Provides Full Memory Protection

The OS as a team of processes

The QNX OS consists of the small Neutrino microkernel managing a group of cooperating processes. As the following illustration shows, the structure looks more like a team than a hierarchy, as several "players" of equal rank interact with each other through the coordinating kernel.

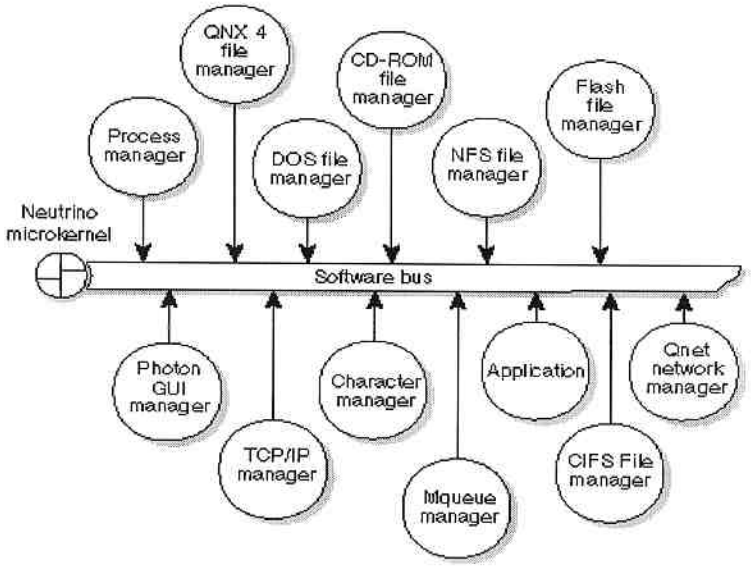


Figure 7.2 The QNX Neutrino OS as a Team of Processes

Interprocess communication

When several threads run concurrently, as in typical realtime multitasking environments, the operating system must provide mechanisms to allow them to communicate with each other.

Interprocess communication (IPC) is the key to designing an application as a set of cooperating processes in which each process handles one well-defined part of the whole.

Message-passing

QNX supports *message passing* as the fundamental means of IPC. In QNX, a message is a parcel of bytes passed from one process to another. The OS attaches no special meaning to the content of a message - the data in a message has meaning for the sender of the message and for its receiver, but for no one else. Besides message passing, Neutrino offers signals, shared memory and pipes for IPC

Scheduling

QNX Neutrino implements thread level scheduling. Each thread is assigned a priority between 1 and 63. The next thread to run scheduled by the scheduler is the *ready* thread having the highest priority (a special *idle* thread in the process manager has a priority of zero and is always ready to run).

QNX Neutrino implements the following scheduling algorithms: FIFO, round robin, adaptive and sporadic scheduling. Each thread in the system may run using any algorithm. They are effective on a per-thread basis, not on a global basis for all threads and processes on a node.

The FIFO and round-robin scheduling algorithms apply only when two or more threads that share the same priority are *ready* (i.e., the threads are directly competing with each other). The adaptive and sporadic methods, however, employ a "budget" for a thread's execution. In all cases, if a higher-priority thread becomes READY, it immediately preempts all *lower-priority* threads

Synchronization

QNX provides the POSIX-standard thread-level synchronization primitives, some of which can even be used between threads in different processes. The synchronization services include the following: mutexes, condvars, barriers, semaphores, and readers/writer locks.

Timers

Neutrino directly provides the full set of POSIX timer functionality. The POSIX timer model is quite rich, providing the ability to have the timer expire on:

- an absolute date
- a relative date (i.e. *n* nanoseconds from now)
- cyclical (i.e. every *n* nanoseconds).

The cyclical mode is very significant, because the most common use of timers tends to be as a periodic source of events to *signal* a thread into life to do some processing and then go back to sleep until the next event. In the cyclic mode the timer is set once and then is automatically reloaded at every expiry.

7.3 The Controller Application

The diagram of the controller application, which utilizes *client/server* architecture is shown in Figure 7.3

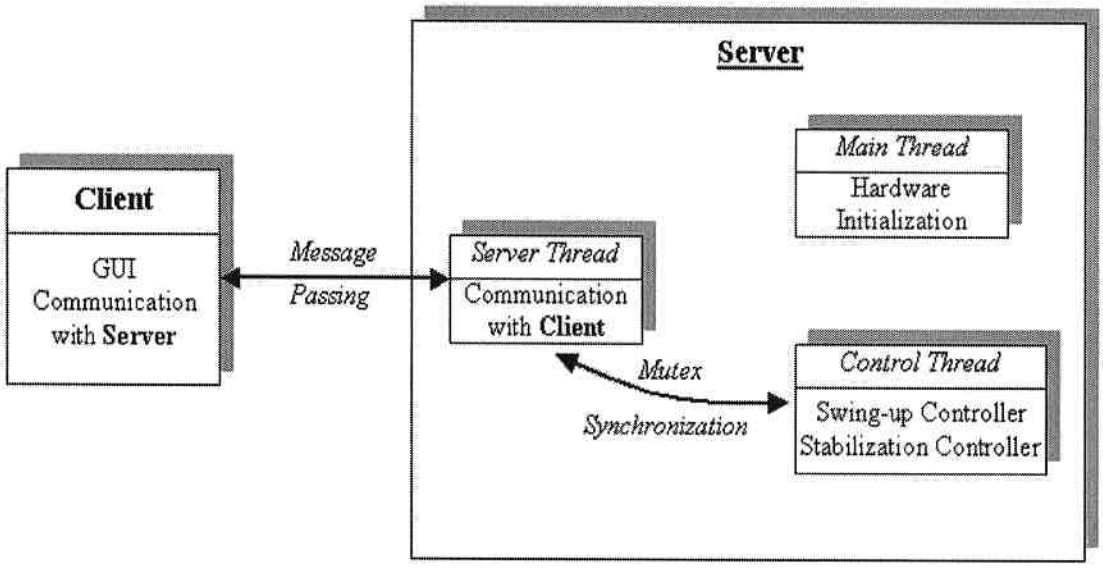


Figure 7.3 Multithreaded Client/Server Application Architecture

Both the Client and Server applications are written entirely in C and run as two independent processes, which communicate between themselves via message passing.

7.3.1 The *Server* Application

The Server is implemented as a multithreaded application. The function of each of the thread is as follows:

Main Thread

The main thread is responsible for initializing the hardware (the DAC and the optical decoder board). After successful initialization of hardware the main thread creates and starts the Server Thread

Server Thread

The Server Thread creates a *message channel* and advertises its *id* in a text file accessible by the client. In its main loop the *Server Thread* listens for messages from the client through the blocking *MsgReceive*(..) call. Two types of message are supported: data command and messages. Currently a *data* message encapsulates the feedback gain vector. Two command messages are implemented. When *Start* message is received from the client the *Server Thread* creates a *timer* structure (the timer is configured to expire every 2 ms and thus provides the base for the required sampling rate) and creates and starts the *Control Thread.* At an *End* request from the client the Server Thread stops the *Control Thread* and the necessary memory releasing operations are performed

Control Thread

The highest possible priority is assigned to the Control Thread. It implements the pendulum swing-up and stabilization controller as well as the switching between them. One time step of the appropriate control function is performed at every expiry of the timer.

A structure that stores the stabilization controller feedback gains is shared between the *Server Thread* and the *Control Thread.* The problem that may occur in this case is that the *Server Thread* can be preempted by the *Control Thread* in the middle of writing the feedback gains. Thus the *Control Thread* could get inconsistent values for the feedback gains. To prevent this scenario, the feedback gain structure is protected by a mutual exclusion lock (*mutex*) and whoever wants to write to or read from the structure has to obtain the mutex first by the system call pthread_mutex_lock(). As blocking of the *Control Thread* is unacceptable it uses the non-blocking system call pthread_mutex_trylock() to try to obtain the mutex.

7.3.2 The *Client* Application

The *Client* Application provides interaction between the user and the system by the means of GUI (Graphical User Interface). The GUI has been built with *Photon Application Builder* IDE.

At user's request to connect to the *Server* the *Client attaches a connection* to the channel advertised by the *Server* message. Further messages are sent to the Server over this connection.

The *Client* application provides the user with the ability to change the closed-loop pole locations of the inverted pendulum system. The stabilizing controller feedback gains are implemented on the base of the newly entered poles by implementing Bass-Gura's formula and encapsulating in a *message* that is sent to the server. (It has been decided that the feedback gains should be calculated in the *Client* application. Equally, the Bass-Guara's formula can be implemented in the Server application (which may be considered by someone a better conceptual approach). In this case, since a delay in the reply message from the server is undesirable it would be better to dedicate a different thread to perform the feedback gains calculations).

7.3 Chapter Conclusions and Recommendations

QNX Neutrino has been chosen as the Operating System platform to run the software control application. This choice has been mainly dictated by the Neutrino's great real time performance providing predictability and reliability.

The actual swing-up sand stabilization controllers are implemented in the server application whereas the client application provides the interface to the user. The flexibility established with the introducing of the client / server architecture can be further utilized. With minimal effort the client can be embedded in a web application and communicate with the server remotely via TCP socket. Thus the pendulum apparatus can be transformed in an ultimate educational tool – control strategies can be demonstrated and tested from a lecture room located virtually anywhere in the world.

8. Conclusions

The general topic of control of under-actuated systems has been approached by studying the performance of control methods on a benchmark pendulum on a cart system.

A pendulum swing up and stabilization controllers were successfully implemented on the experimental system. For the purpose of the pendulum swing nonlinear state-space controller, fuzzy controller and embedded artificial constraints (EAC) controller up were designed and simulated. A linear state-space controller for stabilization of the pendulum in its inverted position was considered to work in conjunction with the fuzzy and the EAC swing up controllers. Experimental swing up and stabilization tests of the EAC swing up controller and the linear state-space stabilizing controller were completed.

8.1 Control Methods Conclusions

8.1.1 Nonlinear State-Space Controller

The main reason for investigating the nonlinear state-space controller is that it considers the exact form of the differential equations describing the system's dynamics. The exact feedback linearization method is very well described and with straightforward derivation. However, the carried calculations and simulations proved that the cart-pole system dynamics do not satisfy the conditions for existence of exact feedback linearization. After analyzing a number of nonlinear methods described in the literature (including partial feedback linearization and approximate linearization) it was decided that a method combining the Lie theoretic approach with linearization with nonlinear output injection was the most suitable candidate for a nonlinear controller. The mathematical description of the method was adapted to fit the pendulum physical model. Optimal pole placement

method was used for the equivalent linear system. The controller is capable of performing both the swing up and the stabilization of the pendulum in its inverted position. Detailed continuous time simulations of the system behavior under the proposed controller were conducted. The simulation showed that the control signal stayed in the acceptable voltage range during both the swing up and stabilization. The limitation that prevented the method of implementation on the pendulum apparatus was that the simulations clearly showed a need of a cart rail of approximately 2.0 m, that length exceeds more than twice the length of the cart rail of the physically available cart-pole system.

8.1.2 Fuzzy Logic and the Pendulum on a Cart System

A fuzzy logic based swing-up controller was investigated and designed. The strength of the fuzzy –logic controller design lies in the ability to conceive controller without having to model the system in terms of its governing differential equations. A human operator based strategy for swinging the pendulum was examined. That strategy suggested that the cart position, the pendulum angle and the pendulum angular velocity were the system states selected as the three fuzzy engine inputs to be fuzzified. A Sugeno type fuzzy engine was designed. The engine consists of 10 *if-then* rules and the antecedent part of each of the rules combines 2 of the 3 fuzzy inputs. The final form of the fuzzy membership functions and the final structure of the rules were chosen after iterative tuning simulations had been performed. The pendulum swing-up/stabilization was simulated by combining the fuzzy swing up controller with the linear state-space stabilizing controller. The simulations showed a satisfactory behavior of system under the fuzzy swing up controller fitting all of the systems physical parameters. The main reason for not attempting physical experiments was that the difficulty for achieving smooth

continuous control signal form the fuzzy engine would increase the demand on the actuator.

8.1.3 Embedded Artificial Control (EAC) for the Inverted Pendulum

The motivation when for initiating the research for the EAC method was to attempt to transform the nonlinear pendulum swing up control problem into a linear controller design.

The analysis showed the form of the pendulum's energy / cart's acceleration relation. The appropriate, smooth, pendulum angle driven cart trajectory, providing the necessary regions of the cart acceleration and deceleration was derived and simulated. Thus the method establishes a link between the system state pendulum angle and cart position by imposing of an artificial constraint. Consequently, the nonlinear control task is reduced into a linear controller design without it being necessary to consider the entire system dynamics. The underlying linear cart position controller was further designed in time domain by utilizing the root-locus method. Both the time and frequency domain behavior of the system with the controller were carefully examined. The simulations proved the feasibility of the concept.

The method was successfully implemented on the physical apparatus. The conducted experiment revealed that the controller is rather robust in a sense that ensures smotth and predictive swinging of the pendulum.

8.1.4 Stabilizing Linear State-Space Controller

The stabilizing controller was realized as a classic linear state-space controller. Optimal gain selection by the solution of the Ricatti equation was explored. Principles for direct

pole placement are also presented. Specifically, it was concluded from experimental investigations that the Ricatti algorithm for optimal pole placement provided the most successful method for obtaining a working control system. The *optimal* design could then be further improved by direct eigenvalue selection. Although this design approach seems to be a contradiction in the sense of what *optimal* is, the optimal design algorithm does in fact provide the most optimal solution of the specified cost function. However, the task of modifying the cost function to provide specific behavior (robustness) is more difficult than the direct refinement of the closed-loop pole locations.

The system was tested for observability. Since direct differentiation would cause deterioration of the controller performance, a linear state observer was implemented to recover the pendulum angular and cart linear velocities

8.2 The Software Implementation

The EAC swing-up and linear state-space stabilizing controllers were implemented on a digital computer running QNX Neutrino 6.1 real-time operating system. The software was written entirely in C and utilizes many of the QNX features. The client/server architecture was utilized for building the application. The server part is responsible for the real-time control. The client part allows the user to change some of the control parameters such as the pole locations of the stabilizing controller.

With minimal effort the client can be embedded in a web application and communicate with the server remotely via TCP socket.

Thus the pendulum apparatus can be transformed in an ultimate educational tool – control strategies can be demonstrated and tested from a lecture room locate virtually anywhere in the world.

8.3 General Recommendations for Control of Under-actuated Systems

The tree presented swing-up control methods can be grouped as methods that require exact system dynamic model and methods that can be applied to uncertain systems.

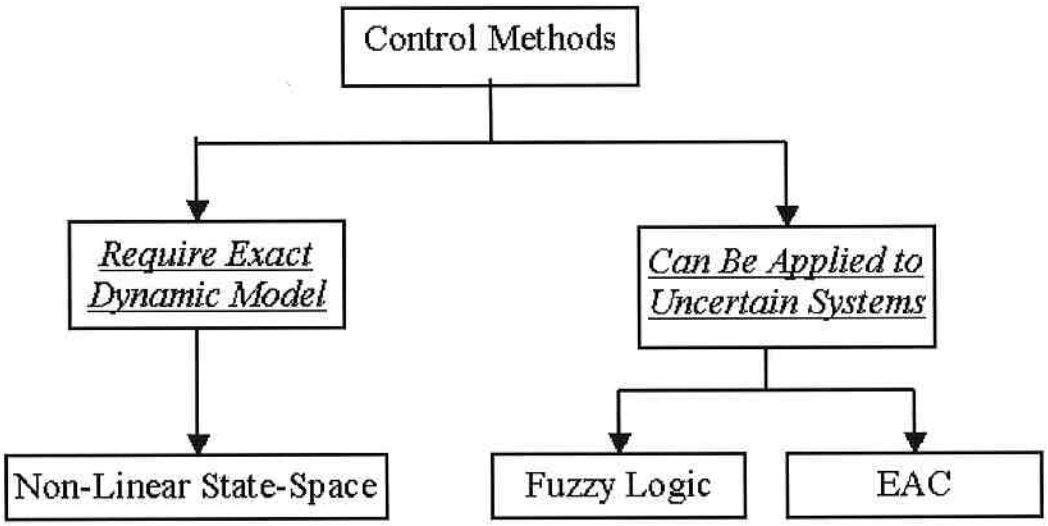


Figure 8.1 Control Method Grouping

The three control methods were evaluated based on their feasibility and performance. The EAC controller proved to be the optimal solution for the pendulum swing-up controller. The experience gained during the pendulum control can be used for establishing a general recommendation for control of under-actuated systems.

Although that the dynamics model of given under-actuated systems may be suitable for developing of a successful nonlinear state-space controller in the general case the methods that can be applied to uncertain systems should be considered. In addition, the EAC controller could be considered as a better approach than the fuzzy logic controller

since it should be easier to develop an artificial state link and then to control linearly instead of building and most importantly tuning a linguistic system model.

8.4 Contributions

Our main contribution we see in the development of controllers for uncertain under-actuated systems. The new developed artificial link approach seems to be most promising for large-scale control of uncertain under-actuated systems.

8.5 Areas for Future Study

The area for future work is quite broad. Two of the possible topics include

- Development of systematic procedure for design of artificial link structures.
- Development of mathematical foundations of stability and performance evaluation of systems with artificial link.

References

1. Absil, P.-A., Sepulchre. "A Hybrid Control Scheme for Swing-up Acrobatics." *Proceedings of the Eurpean Control Conference* Sep 4-7 2001: 35-67.

2. Acosta, A., Aracil. "Nonlinear Control Strategies for the Furata Pendulum." *Control of Intelligent Systems* vol. 30, No. 2 (2002).

3. Astrom, K., Furata. "Swinging Up a Pendulum by Energy Control." *Proc. of 13th Congress of the IFAC* May 1996: pp. 683-88.

4. Baumann, W., Rugh. "Feedback Control of Nonlinear Systems by Extended Linearization." *IEEE Transactions on Automatic Control* vol.1, No. 1 (1986): 40-46.

5. Bedrossian, N. "Approximate Feedback Linearization: The Cart-Pole Example." *Proc. of the 1992 IEEE International Conference on Robotics and Automation* [Nice, France] May 1992: 1987-92.

6. Burden, R., Faire. *Numerical Analysis, 6th Edition.* Brooks/Cole Publishing Co., Pacific Grove, 1997.

7. Chung, C, Hauser. "Nonlinear Control of a Swinging Pendulum." *Automatica* vol. 31, No. 6 (1995): pp. 851-62.

8. Dedrossian, N. "Nonlinear Control of an Under-Actuated Two Link Manipulator." *Proceedings of the 32nd Conference on Decision and Control* (1993): 2234-38.

9. Dorf, R. *Modern Control Systems, 4th Edition.* Addison-Wesley Publishing Co., Inc., 1986.

10. Eastwood, E. "Control Theory and the Engineer." *Proceedings of IEE* Vol, 115, No. 1 (1968): 203-11.

11. Erlic, M., Lu. "Manipulator Control with an Exponentially Stable Velocity Observer." *Proc. American Control Conf.* June 1992: pp. 1241-42.

12. ---. "A Reduced Order Adaptive Velocity Observer for Manipulator Control." *Proc. IEEE Int. Conf. Robotics Automation* [Chicago] vol. 2 (May 1993): pp. 328-33.

13. Fierro R., Lewis F., Lowe. "Hybrid Control for a Class of Underactuated Mechanical Systems." *IEEE Transactions on System, Man, and Cybernetics--Part A: System and Humans* Vol. 29, No.6 (1999).

14. Franklin G., Powell, J. *Digital Control of Dynamic Systems, 3rd Edition.* Addison-Wesley Publishing Co., Inc., Reading, Mass., 1998.

15. Franklin, G., Powel. *Feedback Control of Dynamic Systems, 3rd Edition.* Addison-Wesley Publishing Co., Inc., Reading, Mass.Addison-Wesley Publishing Co., Inc., Reading, Mass., 1994.

16. Friedland, B. *Control System Design.* McGraw-Hill, Inc., New York, 1986.

17. Harbison, S, Stele. *C a Reference Manual, 5th Edition.* Prentice-Hall, Inc., New Jersey, 2002.

18. Hauser, J., Shankar S. "Nonlinear Control Via Approximate Input-Output Linearization: The Ball and the Beam Example." *Proceedings of the 28th Conference on Decision and Control* [Tampa, Florida] Dec. 1989: pp. 1987-93.

19. Henders, M., Soudack. "'In-the-Large' Behavior of an Inverted Pendulum with Linear Stabilization." *International Journal on Non-Linear Mechanics* Vol. 27 (1992): 129-38.

20. Hibbeler, R. *Engineering Mechanics Dynamics, 8th Edition.* Prentice-Hall, Inc., New Jersey, 1998.

21. Isidori, A. *Nonlinear Control Systems, 2nd Edition.* Springer-Verlag, Berlin, 1989.

22. Jamshidi, M., editor. *Applications of Fuzzy Logic*. Prentice-Hall, Inc., New Jersey, 1997.

23. Kawaji, S., Maeda. "Fuzzy Servo Control System for an Inverted Pendulum." *Fuzzy Engineering Toward Human Friendly Systems* (1992): 812-23.

24. Kawashima, T. "Swing-up and Stabilization of Inverted Pendulum Using Only One Sliding Mode Controller with Nonlinear Model Observer." *Third International Conference on Motion and Vibration Control* (1996): 449-54.

25. Khalil, H. *Nonlinear Systems*. Macmallian Publishing Co., New York, 1992.

26. King, P., Mamdani. "The Application of Fuzzy Control Systems to Industrial Process." *Automatica* vol. 13, No.3 (1977): 235-42.

27. Kobayashi, M., Shihohara. "Swing-up and Stabilization of an Inverted Pendulum with Adaptive Indentifier." *International Journal of Control* (1992): 3603-13.

28. Kosko, B. *Fuzzy Engineering*. Prentice-Hall, Inc., New Jersey, 1997.

29. Krener, A. "Approximate Linearization by State Feedback and Coordinate Change." *Systems and Control Letters* vol. 5 (Dec. 1984): pp. 181-85.

30. ---. "Linearization by Output Injection and Nonlinear Observers." *Systems and Control Letters* vol. 3 (June 1983): 47-52.

31. Krener, A., Karahan. "Higher Order Linear Approximations to Nonlinear Control Systems." *Proc. of the 26th Conference on Decision and Control* [Los Angeles, CA] Dec. 1987: 519-22.

32. Krten, R. *Getting Started with QNX Neutrino 2*. Parse Software Devices, Kanata, 2001.

33. Kwon, Y.-H., Kim. "Swin Up Controller for Inverted Pendulum System." *Proceedintgs of the 32nd International Symposium on Robotics* April 19-21 2001: pp. 871-76.

34. Leondes, C., editor. *Fuzzy Theory Systems Techniques and Applications, Volume 2*. Accademic Press, San Diego, 1999.

35. Luenenberger, D. "Observers for Multivariable Systems." *IEEE Transactions on Automatic Control* vol. ac-II (1966).

36. Mamdani, E. "Applications of Fuzzy Algorithms for Simple Dynamic Plant." *Proc. IEE* vol. 121 (1974): 1585-88.

37. Melin, C., Vidolov B. "A Fuzzy PD-Like Scheme for Two Underactuated Planar Mechanisms." *Proc. IEEE International Fuzzy Systems Conference* (2000): 146-53.

38. Middleton, R., Goodwin. *Digital Control and Estimation A Unified Approach.* Prentice-Hall, Inc., New Jersey, 1990.

39. Mowat, R. "Comperative Experimental Study of Various Control Methods in Application to Unstable, Nonlinear Lagrangian Systems." Masc Thesis, Department of mechanical Engineering, University of Victoria, 1994.

40. Novak, V. *Fuzzy Sets and Their Applications*. Adam Hilger, Bristol, 1989.

41. Ogata, K. *Modern Control Engineering, 3rd Edition*. Prentice-Hall, Inc., New Jersey, 1997.

42. Ohsumi, A., Izumikawa. "Nonlinear Control of Swing-Up and Stabilization of an Inverted Pendulum." *Proceedings of the 34th Conference on Decision & Control* [New Orleans, LA] Dec. 1997: pp. 3873-80.

43. Raven, F. *Automatic Control Engineering*. McGraw-Hill, Inc., New York, 1987.

44. Reznik, L. *Fuzzy Controllers*. Newnes, Oxford, 1997.

45. Sain, M., Yurkovich. "Controller Scheduling: A Possible Algebraic Viepoint." *Proc. American Control Conference* (1982): 261-69.

46. Sanchez E., Nuno L., Hsu Y.-C. "Fuzzy PD Scheme for Underactuated Robot

Swing-up Control." *Proc. IEEE International Fuzzy Systems Conference* (1998): 302-06.

47. Sedgewick, R. *Algorithms in C, 3rd Edition*. Addison-Wesley Publishing Co., Inc., Reading, Mass., 1999.

48. Shaw, I. *Fuzzy Control of Industrial Systems*. Kluwer Academic Publishers, Boston, 1998.

49. Song, F., Samuel S. "A Takagi-Sugeno Fuzzy Logic Controller with Only 3 Rules for an l Inverted Pendulum System," 2002.

50. Spong, M. "`The Swing Up Control Problem for the Acrobot." *IEEE Control Systems Magazine* Vol. 15, No. 1 (Feb. 1995): 49-55.

51. ---. "The Control of Underactuated Mechanical System." *Plenary Address at The First International Conference on Mechatronics* [Mexico City] Jan 26-29 1994.

52. ---. "Energy Based Control of a Class of Underactuated Mechanical Systems." *Proc. 13th IFAC World Congress* [San Francisco, CA] Vol. F (July 1996): 431-36.

53. ---. "Modeling and Control of Elastic Joint Robots." *Transactions of ASME* v. 109 (Dec. 1987): pp. 310-19.

54. ---. "Partial Feedback Linearization of Underactuated Mechanical Systems." *IROS'94* [Munich, Germany] Sept 1994: 314-21.

55. ---. "Underactuated Mechanical Systems." *International Workshop on Control Problems in Robotics and Automation* [San Diego, CA] (1997).

56. Spong, M., Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, Inc., New York, 1989.

57. Stimac, A. "Standup and Stabilization of the Inverted Pendulum." *Department of Mechanical Engineering, MIT* (1999).

58. Su, C.-Y. "Adaptive Variable Structure Control of Underactuated Robots." *IEEE Transactons on Automatic Control* Vol. 44, No. 11 (Nov. 1999): 2090-93.

59. Su, C.-Y., Stepanenko. "Adaptive Sliding Mode Control of Robot Manipulators: General Sliding Manifold Case." *Automatica* vol. 30 (1994): pp. 1497-500.

60. Takagi, T., Sugeno. "Fuzzy Idenfication of Systems and Its Applications Modeling and Control." *IEEE Transactions on Systems, Man and Cyberntics* Vol 15, No.1 (1985): 116-32.

61. Teixeira, M., Zak. "Stabilizing Controller Design for Uncertain Nonlinear Systems Using Fuzzy Models." *IEEE Transactions on Fuzzy Systems* Vol. 7, No. 2 (April 1999): 133-42.

62. Vidyasagar, M. *Nonlinear System Analysis, 2nd Edition*. Prentice-Hall, Inc., New Jersey, 1993.

63. Watts, J. "Control of an Inverted Pendulum." *ASEE Annual Conference Proceedings* (1984): 706-10.

64. Wei, Q., Dayawansa. "Nonlinear Control for an Inverted Pendulum Having Restricted Travel." *Automatica* vol. 31, No. 6 (1995): 841-50.

65. Williams, J. *Fundamentals of Applied Dynamics*. John Wiley & Sons, Inc., New York, 1996.

66. Yamakawa, T. "Stabilization of an Inverted Pendulum by a High Speed Fuzzy Logic Controller Hardware Systrem." *Fuzzy Sets and Systems* vol. 32 (1989): 161-80.

67. Yasunobu, S., Mori. "Swing up Fuzzy Controller for Inverted Pendulum Based on a Human Control Strategy." *Proc. IEEE International Fuzzy Systems Conference* (1997): 1621-25.

68. Yi, J., Yubazaki. "Systematically Constructing Stabilization Fuzzy Controllers for Single and Double Pendulum Systems." *Proc. IEEE International Fuzzy Systems Conference* (2000).

69. Yoshida, K. "Swing-Up Control of an Inverted Pendulum by Energy-Based Methods." *Proc. of the American Control Conference* June 1999: pp. 4045-47.

70. Zadeh, L. "Fuzzy Algorithm." *Information and Control* Vol. 12 (1968): 94-102.

71. ---. "Outline of a New Approach to the Analysis of Complex Systems and Descision Process." *IEEE Transactions Systems, Man and Cybernetics* Vol. 3 (1973): 28-44.

72. ---. "A Rationale for Fuzzy Control." *Trnsactions Asme, Journal of Dynamic Systems Measurement and Control* vol. 94 (1972): 3-4.

Appendix A

The Equations of Motion for the DC Motor

The following presentation is motivated by the need to find the horizontal force F_u acting on the cart and generated by the DC motor.

The linear amplifier is capable of generating ±22 A at ±27 volts. To ensure that the amplifier works in the linear range of its V/A characteristic curve (i.e., it never saturates), the governing electrical characteristic of the DC motor/amplifier system will also be presented.

The horizontal force acting on the cart F_u can be related to the torque τ generated by the motor via the effective gear radius r :

$$F_u = \frac{\tau}{r} \tag{A.1}$$

If both the amplifier and the DC motor are never saturated, then the relation between the torque generated by the permanent magnet DC motor and the armature current ia is given by the linear dependence:

$$\tau = T_t i_a \tag{A.2}$$

where T_t is the linear torque constant for the motor.

As the amplifier works in linear mode the armature current is proportional to the input voltage:

$$i_a = uA_G \tag{A.3}$$

where AG is the amplifier's gain.

Combining A.1 and A.3 the force acting on the cart is given by:

$$F_u = \frac{A_G T_t u}{r} \tag{A.4}$$

Equation A.4 will be referred to as the equation of motion for the DC motor, i.e the force on the cart due to the DC motor. Note that equation A.4 assumes that the operation of the DC motor is maintained within its linear range as well as the amplifier is never saturated. To insure that the amplifier is never saturated, one must model the electrical characteristics of the motor. This modeling is presented in the next section.

Deriving the Relationship Between Input Voltage and Motor Voltage

With reference to the definitions of variables and constants given in Rable A.1 and by utilizing the Kirchoff's Voltage Law for the loop shown in Figure A-1, the motor input voltage (the armature voltage) is:

$$U_m = R_a i_a + L_a \frac{di_a}{dt} + Bemf \tag{A.5}$$

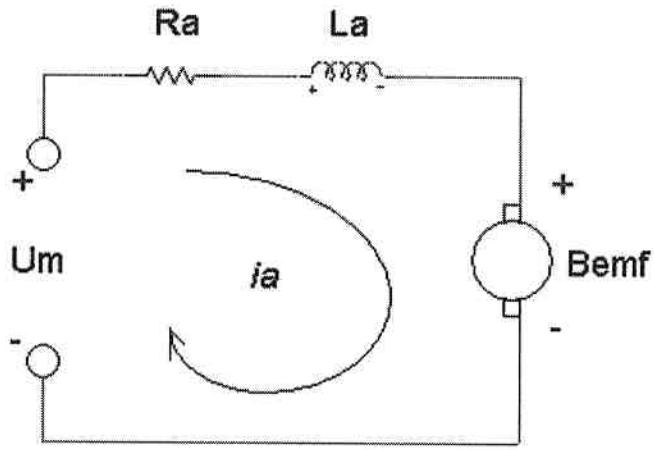


Figure A.1 DC Motor Electrical Diagram

Since the armature current is proportional to the input voltage u

$$i_a = A_G u \tag{A.6}$$

and the back EMF of the DC motor is given as

$$Bemf = E\dot{\theta}_m \tag{A.7}$$

Equation (A.5) can be expressed in terms of input control voltage rather than armature current:

$$U_m = R_a A_G u + L_a \frac{d(A_G u)}{dt} + E\dot{\theta}_m \tag{A.8}$$

The angular velocity of the motor, $\dot{\theta}_m$, can be related to the horizontal velocity of the cart $\dot{x}$ by

$$\dot{\theta}_m = \frac{\dot{x}_M}{r_m} \tag{A.9}$$

where r_m is the radius of the gear attached to the motor. Thus (A.8) becomes:

$$U_m = R_a A_G u + L_a \frac{d(A_G u)}{dt} + E\frac{\dot{x}_M}{r_m} \tag{A.10}$$

Equation (A.10) represents the voltage across the motor due to an input voltage u.

Since the electrical time constant of the motor is short compared to its physical (or mechanical) time constant (i.e., the current in the motor reaches a maximum value much faster than the angular velocity of the motor reaches a maximum) the derivative term in the above relation can be ignored, giving

$$U_m = R_a A_G u + E\frac{\dot{x}_M}{r_m} \tag{A.11}$$

Equation (A.11) must be used in system simulations to insure that the voltage across the motor never exceeds ±27V:

$$|U_m| \leq 27V \tag{A.12}$$

τ	Torque generated by motor	Nm
T_t	Motor torque constant	N·m/A
F_u	Force acting on cart	N
$Bemf$	Back EMF of motor	V
E	Back EMF constant	V·s/rad
L_a	Armature inductance	Henry
R_a	The height of the cylinder at the end of the rod	Ohms
U_m	Input voltage to the motor	V
u	Control input voltage to the amplifier	V
A_G	Amplifier gain	A/V
i_a	Armature current	A
r_m	Motor gear radius	m
$\dot{\theta}_m$	Motor angular velocity	m/s^2
$\dot{x}_M$	The horizontal velocity of the cart	m/s

Table A.1 DC motor parameters

Appendix B

Equations of Motion for the pendulum on a cart system

The derivation of the equations of motion for pendulum on a cart utilizes the method of Lagrangian Dynamics [65] to find a set of coupled second-order nonlinear differential equations describing the relationship between the input voltage to the amplifier, the horizontal position of the cart and the angular position of the pendulum.

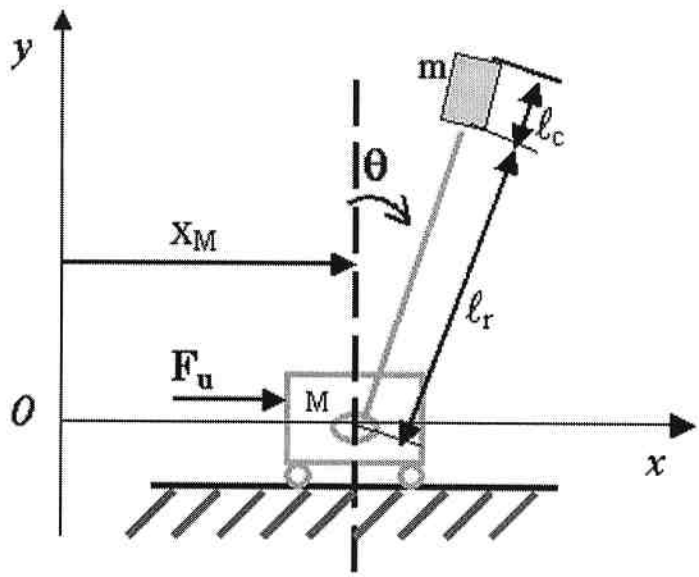


Figure B.1 Pendulum on a cart schematics

Lagrangian Dynamics

Lagrangian Dynamics refers to the determination of motion by the Lagrange formula

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = \Xi_j \tag{B.1}$$

where

q_j - generalized coordinates

Ξ_j - generalized forces

L = T-V, and defined as Lagrangian

T = $f(q_j, \dot{q}_j)$, kinetic energy of the total system

V = $f(q_j)$, potential energy of the system

Generalized Coordinates

With reference to figure B.1 the pendulum/cart system has two degrees of freedom and can therefore be fully represented using two generalized coordinates. The generalized coordinates are chosen as the horizontal displacement of the cart x_M and the angular displacement of the pendulum θ:

$$q_j : x_M, \theta \tag{B.2}$$

A complete and independent set of the associated admissible variations is given by:

$$\delta q_j : \delta x_M, \delta\theta \tag{B.3}$$

so the system is holonomic and the Lagrange's method can be applied.

Generalized Forces

An expression for the generalized forces, x_j, can be derived from the nonconservative work, given by:

$$\delta W^{nc} = \sum_{i=1}^{n} \mathbf{f}_i^{nc} \bullet \delta \mathbf{R}_i \tag{B.4}$$

where $\mathbf{R}_i$ is the position vector where the ith nonconservative force acts. The nonconservative forces for the cart pendulum system result from the input force and the viscous friction terms:

$$\delta W^{nc} = Fu\delta x_M - b_c \dot{x}_M \delta x_M - b_p \dot{\theta}\delta\theta \tag{B.5}$$

The equations for the generalized forces can be derived, by comparing equations (B.4) and (B.5) and with reference to Equation (A.4)

$$\Xi_{x_M} = F_u - b_c \dot{x}_M = \frac{A_G T_t}{r} u - b_c \dot{x}_M \tag{B.6}$$

$$\Xi_\theta = -b_p \dot{\theta} \tag{B.7}$$

Potential and Kinetic Energy Functions

As the cart moves only in horizontal direction the potential energy of the system is determined entirely by the angle of the pendulum:

$$V = mgl_g \cos\theta \tag{B.8}$$

where m is the total mass of the pendulum rod and cylinder

$$m = m_r + m_c \tag{B.9}$$

and l_g is the distance from the pendulum pivot to the center of mass of the pendulum (rod and cylinder) :

$$l_g = \frac{\frac{1}{2}m_r l_r + m_c(l_r + l_c/2)}{m_r + m_c} \tag{B.10}$$

The kinetic energy of the entire apparatus is:

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}mv_g^2 + \frac{1}{2}J_p\dot{\theta}^2 \tag{B.11}$$

Where M is the total mass of the cart (including the equivalent mass for the motor inertia) and J_p is the mass moment inertia of the pendulum rod and cylinder about their center of mass:

$$J_p = \left(J_e + m_g l_g^2\right) + \left(\frac{1}{12} m_r l_r^2 + m_r \left(l_g - \frac{1}{2} l_r\right)^2\right) + \\ + \left(\frac{1}{12} m_c \left(3r_c^2 + l_c^2\right) + m_c \left(\left(l_r + l_c/2\right) - l_g\right)^2\right) \tag{B.12}$$

with J_e the mass moment of inertia of of the encoder and encoder shaft with total mass m_e.

The velocity of the center of mass of the pendulum v_g, can be found by summing the horizontal and vertical components of velocity. Thus, with reference to Figure B.2, v_g^2 is given by:

$$v_g^2 = \left[\frac{d}{dt}\left(x + l_g \sin\theta\right)\right]^2 + \left[\frac{d}{dt}\left(l_g \cos\theta\right)\right]^2 =$$

$$= \left[\left(\dot{x} + l_g \dot{\theta} \cos(\theta)\right)\right]^2 + \left[-l_g \dot{\theta} \sin\theta\right]^2 =$$

$$= \dot{x}^2 + 2 l_g \dot{x} \dot{\theta} \cos\theta + l_g^2 \dot{\theta}^2 \tag{B.13}$$

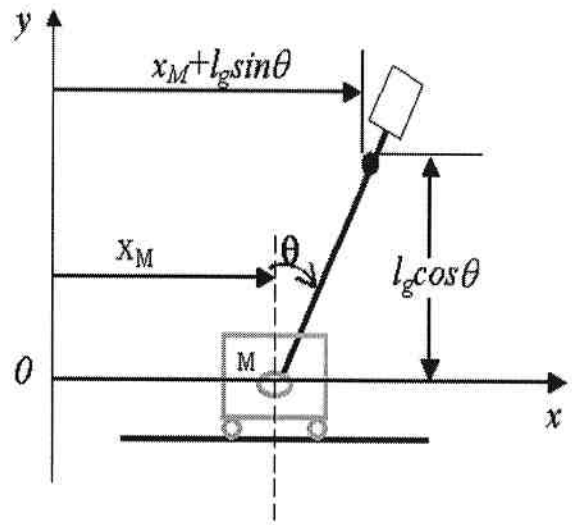


Figure B.2 Finding the velocity of the Pendulum Rod Center of Mass

Thus, the Lagrangian of the system is:

$$L = T - V = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{x}^2 + 2l_g\dot{x}\dot{\theta}\cos\theta + l_g^2\dot{\theta}^2\right) + \frac{1}{2}J_p\dot{\theta}^2 - mgl_g\cos\theta \qquad \text{(B.14)}$$

Lagrange's Equations

With reference to (B.1) the Lagrange equation for with respect to x_M and θ are:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_M}\right) - \frac{\partial L}{\partial x_M} = \Xi_{x_M} \qquad \text{(B.15)}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right)-\frac{\partial L}{\partial \theta}=\Xi_{\theta} \tag{B.16}$$

The partial derivatives can be derived by differentiating (B.14)

$$\frac{\partial L}{\partial \dot{x}_M}=(M+m)\dot{x}_M+ml_g\dot{\theta}\cos\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_M}\right)=(M+m)\ddot{x}_M+ml_g\ddot{\theta}\cos\theta-ml_g\dot{\theta}^2\sin\theta \tag{B.17}$$

$$\frac{\partial L}{\partial x_M}=0$$

$$\frac{\partial L}{\partial \dot{\theta}}=ml_g\dot{x}\cos\theta+\left(ml_g^2+J_p\right)\dot{\theta}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right)=ml_g\left(\ddot{x}\cos\theta-\dot{x}\dot{\theta}\sin\theta\right)+\left(ml_g^2+J_p\right)\ddot{\theta} \tag{B.18}$$

$$\frac{\partial L}{\partial \theta}=mgl_g\sin\theta-ml_g\dot{x}\dot{\theta}\sin\theta$$

Substituting (B.6) and (B.17) in (B.15) and (B7) and (B.18) in (B.16) the two equations of motion become:

$$\begin{cases} (M+m)\ddot{x}_M + ml_g \cos\theta\ddot{\theta} - ml_g\dot{\theta}^2 \sin\theta + b_c\dot{x}_M = \dfrac{A_G T_t}{r}u \\ \left(ml_g^2 + J_p\right)\ddot{\theta} + ml_g\ddot{x}_M \cos\theta - mgl_g \sin\theta + b_p\dot{\theta} = 0 \end{cases} \tag{B.19}$$

M	The total effective mass of the cart	kg
m_r	The mass of the pendulum rod	kg
m_c	The mass of the cylinder at the end of the rod	kg
m	Total mass of pendulum	kg
m_e	The mass of the encoder disk and shaft for pendulum	kg
l_r	The length of the pendulum rod	m
l_c	The height of the cylinder at the end of the rod	m
l_g	The distance from the rod pivot to the center of mass of pendulum	m
J_p	The moment of inertia of the pendulum about its mass center	kg·m^2
b_c	The viscous friction coefficient for horizontal motion of the cart	N·s/m
b_p	The viscous friction coefficient for the angular motion of rod	N·s
r	The effective gear radius of transmission	m
g	Acceleration due to gravity	m/s^2
A_G	The voltage controlled current amplifier gain	A/V

T_t	The toque constant of the DC motor	N·m/A
x_M	The horizontal position of the cart	m
$\dot{x}_M$	The horizontal velocity of the cart	m/s
θ	The angular position of the pendulum	rad
$\dot{\theta}$	The angular velocity of the pendulum	rad/s
u	The input voltage to the amplifier	V

Table B.1 Definitions of cart-pole system parameters

Appendix C

Physical Values of System Parameters

Appendix C presents the physical values of the system parameters used in the simulations and experiments. The parameter identification has been performed by Ryan Mowat in his master's thesis [39].

M	The total effective mass of the cart	0.8093 kg
m_r	The mass of the pendulum rod	0.068 kg
m_c	The mass of the cylinder at the end of the rod	0.146 kg
m	Total mass of pendulum	0.2140 kg
m_e	The mass of the encoder disk and shaft for pendulum	0.025 kg
l_r	The length of the pendulum rod	0.28 m
l_c	The height of the cylinder at the end of the rod	0.04 m
l_g	The distance from the rod pivot to the center of mass of pendulum	0.2523 m
J_p	The moment of inertia of the pendulum about its mass center	4.3e-3 kg·m^2
b_c	The viscous friction coefficient for horizontal motion of the cart	2.4 N·s/m
b_p	The viscous friction coefficient for the angular motion of rod	1e-3 N·s
r	The effective gear radius of transmission	0.0254 m
g	Acceleration due to gravity	9.81m/s^2
A_G	The voltage controlled current amplifier gain	2.2 A/V

T_t	The toque constant of the DC motor	0.0635 N·m/A
E	Back EMF constant	0.0678 V·s/rad
L_a	Armature inductance	3.0 mH
R_a	The height of the cylinder at the end of the rod	1.63 Ohms

Table C.1 Physical values of the cart-pole system parameters

Appendix D

Matlab Simulation Files

Appendix D contains the Matlab script files used for the simulations presented in the Thesis.

Script Title: pend_const.m

```
%-------------------------------------------%
%
% File: pend_cost
% Contains system constants used in
% all of the control method simulations
% August 2003
% Dilian Stoikov
% -------------------------------------------%
%
% Index:
% Ra - armature resistance, [Ohms]
% Tt - torque constant; [N*m/A]
% E - back EMF constant; [V*s/rad]
% Jm - effective moment of inertia for motor and pulleys; [kg*m^2]
% Ag - amplifier gain; [A/V]
% r1, r2, r3, rp, pulley radius, from motor to idler pulley; [m]
% r - effective pulley radius [m]
% bc - visous friction for cart pulley system; [N*s/m]
% bp - viscous friction for pendulum; [N*s/m]
% l - pendulum length; [m]
% mr - mass of pendulum rod; [kg]
% mc - mass of cylinder at the end of pendulum rod
% m - total mass rod + cylinder; [kg]
% lg - center of gravity of rod/cylinder
% g - acceleration due to gravity; [m/s^2]
% M - effective mass of cart; [kg]
% Jp - Moment of inertia of the pendulum about its mass center [kg*m^2]
%
%-------------------------------------------%

Ra = 1.63;
Tt = 0.0635;
E = 7.1e-3*60/(2*pi);
Jm=0.06004261738608;
Ag = 2.2;
```

```
r1 = 2.00*0.5*0.0254;
r2 = 1.00*0.5*0.0254;
r3 = 1.00*0.5*0.0254;
rp = 1.00*0.5*0.0254;
r = r1*r3/r2;

bc = 2.4;
bp = 0.001;

%rod chars
l = 0.30;
lc=0.04;
r_c=1.27e-2;
mr= 0.068;
mc = .146;
m=mr+mc;


%effective mass of cart
mwheels = 0.040;
M = 0.230 + 0.390 + 2*mwheels+ ((0.02835)*0.21*(85.14/12+(r2/r3)^2*14.137/12)) + Jm;


g = 9.81;
V = 35;

lg = (mr*l/2 + mc*l)/(mr+ mc);
Jp = (0.4e-7+(0.003+mwheels)*lg^2)+(1/12*mr*l^2+mr*(lg-l/2)^2) +
      (1/12*mc(3*r_c^2+lc^2)+mc*(l-lg)^2);
```

D1 Scripts used for the simulations in Chapter 3

Script Title: nonlinear_controller.m

```
%---------------------%
%File: nonlinear_controller.m
%continues time simulations of the nonlinear
%state space controller
%August 2003
%Dilian Stoikov
%---------------------%
global K B  M0 J0

pend_const %load system physical parameters

T=0.02;
```

```
Time=10;
M0 = m+M;
J0 = Jp+m*lg^2;
t=0:T:Time;
first=1;
first1=1;


%-------------------%
%state-space model of the linearized system
%and controller derivation

A = [0 0 1 0;
    0 0 0 1;
    0 1 0 0;
    0 0 0 -1/J0*bp];

B=[0; 0; 1; -m*l/J0];

Q = diag([1 1000 1 0.1]); % best one swing
R=5;
K = lqr(A,B,Q,R);
%poles of the closed-loop linear system
Ac = (A-B*K);
poles = eig(Ac);


%-------------------%

%options = odeset('OutputFcn',@odeout1);
y0=[0; pi; 0; 0]; %initial conditions
[t y]=ode45(@ode_nonlin,[0 Time], y0); %use ODE solver for the state vector
x=y(:,1);
x_dot=y(:,3);
theta=y(:,2);%*180/pi;
theta_dot = y(:,4);

%-----------------%



%------------------------%
%recover control signal from
%the values of the state vector


for k=1:length(theta)-1
```

```
    py=[0; 0; 0; (-1/J0*m*l*(theta(k)*cos(theta(k))-g*sin(theta(k))))];
    ry=[0; 0; 1; (-1/J0*m*l*cos(theta(k)))];
    pmy = [0; 0; -2/pi; ((2*pi/J0)*m*l*cos(theta(k)))];

    f3 = (((Jp+m*l^2)*(m*l*theta_dot(k)^2*sin(theta(k))-bc*x_dot(k)))-
          (m*l*cos(theta(k))*(m*g*l*sin(theta(k))-bp*theta_dot(k))))/((M+m)*(Jp+m*l^2)-
                m^2*l^2*(cos(theta(k))^2));

    f4 = (-m*l*cos(theta(k))*f3+m*g*l*sin(theta(k))-bp*theta_dot(k))/((Jp+m*l^2));
    g3 = (Ag*Tt/r)*(Jp+m*l^2)/((M+m)*(Jp+m*l^2)-m^2*l^2*(cos(theta(k))^2));

    p0y = py;
    alpha = theta(k);
    if and(theta(k)>65*pi/180,theta_dot(k)>0)
       p0y = py+pmy;
       alpha = theta(k)-2*pi;
    end
    x_n = [x(k); alpha; x_dot(k); theta_dot(k)];
    vWave1 = - K*x_n;
    v0=1/(ry'*ry)*ry'*(B*vWave1-p0y);
    v=v0+theta(k);
    u(k)=-f3/g3+1/g3*v;
end
u(k+1)=u(k);


%-----------------------%

figure(1)
plot(t,theta*180/pi,'-',t,x*100,'-.');
title('Response of the Cart Position and Pendulum Angle');
xlabel('Time [sec]');
ylabel('Pendulum Angle [deg], Cart Position [cm]');
legend('Pendulum Angle', 'Cart Position');
figure(2)
plot(t,u);
title('Control Signal');
xlabel('Time [sec]');
ylabel('Input Voltage u(t) [V]');
```

Script Title: ode_nonlin.m

```
%------------------------------%
% File: ode_nonlin.m
% deifines the system differential equations
% August 2003
% Dilian Stoikov
```

```
%-----------------------------%
function dydt=odefn(t,y);

global K B M0 J0
pend_const; %load system parameters

x=y(1);
theta=y(2);
d_x=y(3);
d_theta=y(4);


py=[0; 0; 0; (-1/J0*m*l*(theta*cos(theta)-g*sin(theta)))];
ry=[0; 0; 1; (-1/J0*m*l*cos(theta))];
pmy = [0; 0; -2/pi; ((2*pi/J0)*m*l*cos(theta))];

f3 = (((Jp+m*l^2)*(m*l*d_theta^2*sin(theta)-bc*d_x))-(m*l*cos(theta)*(m*g*l*sin(theta)-
          bp*d_theta)))/((M+m)*(Jp+m*l^2)-m^2*l^2*(cos(theta)^2));
f4 = (-m*l*cos(theta)*f3+m*g*l*sin(theta)-bp*d_theta)/((Jp+m*l^2));
g3 = (Ag*Tt/r)*(Jp+m*l^2)/((M+m)*(Jp+m*l^2)-m^2*l^2*(cos(theta)^2));
g4 = -m*l*cos(theta)*g3/(Jp+m*l^2);

p0y = py;
alpha = theta;
if and(theta>50*pi/180,d_theta>0)
   p0y = py+pmy;
   alpha = theta-2*pi;
end

%--------------
%calculate control action
x_n = [x; alpha; d_x; d_theta];
vWave = - K*x_n;%linear controller
v0=1/(ry'*ry)*ry'*(B*vWave-p0y);
v=v0+theta;
u=-f3/g3+1/g3*v;
%-----------

dd_x=f3+g3*u;
dd_theta=f4+g4*u;


dydt(1)=d_x;
dydt(2)=d_theta;
dydt(3)=dd_x;
dydt(4)=dd_theta;
dydt=dydt';
```

D2 Scripts used for the simulations in Chapter 4

Script Title: fuzzy.m

```
%-----------------------------%
% File: fuzzy.m
% continuous  time simulations of
% the fuzzy controller for
% swing up and linear state-space
% controller for system stabilization
%
% August 2003
% Dilian Stoikov
%-----------------------------%

global M0 J0 G a first x_ref stab stab_first x_ref_stab
pend_const %load system physical parameters

T=0.02;
Time=7.1;
M0 = M+m;
J0 = Jp+m*lg^2;

%---------------------------%
%global variables definitions
first=1;
stab_first=1;
x_ref=0;
first=1;
stab=0;
stab_first=1;
x_ref_stab=0;
%bc=35;
%--------------------------%


%--------------------------%
%derivations for stablizing linear
%state space controller

M_mat = [1 0 0 0;
    0 1 0 0;
    0 0 (M+m) (m*lg);
    0 0 (m*lg) (m*lg^2+Jp)];
```

```
K = [0 0 1 0;
     0 0 0 1;
     0 0  -bc 0;
     0 (m*g*lg) 0 -bp];

Q = [0;0;Ag*Tt/r;0];

%state space form
A = inv(M_mat)*K;
B=inv(M_mat)*Q;
Q_opt = diag([3e6 7*180/pi 0.1 1]);
optR = 1000;
G = lqr(A,B,Q_opt,optR);

%--------------------------


a=readfis('fuzzy_model_up4.fis');%obtain the fuzzy model

t=0:T:Time;
l=length(t);



y0=[0; 40*pi/180; 0; 0]; %initial system conditions
[t y]=ode45(@ode_fuzzy,[0 Time], y0,options); %use ODE solver
x=y(:,1);
x_dot=y(:,3);
theta=y(:,2);
theta_dot = y(:,4);



%------------------------%
%recover control signal from
%the values of the state vector

first=1;
stab_first=1;
stab=0;
x_s(1)=0;
u(1)=0;
for i=2:length(theta)-1
 if or((180-theta(i)*180/pi)<4, stab==1)
   stab=1;
    if stab_first==1
      x_ref_stab=x(i);
      stab_first=0;
    end
    theta(i)=-(pi - theta(i))*sign(theta(i));
```

```
      xx1=x(i)-x_ref_stab;
      x_n=[xx1, theta(i), x_dot(i), theta_dot(i)]';
      u(i)=-G*x_n;
      theta(i)=  pi + theta(i);
      x_s(i)=x(i);
    else
      if (abs(theta_dot(i))<0.5)
       theta_dot(i)=-sign(theta(i))*0.2;
      end
      if theta(i)*theta_dot(i)<=0
       u(i)=0;
       first=1;
       x_s(i)=x_s(i-1);
      else
       if first==1
        first=0;
        x_ref=x(i);
       end
       u(i)= evalfis([abs(theta(i)*180/pi) abs(theta_dot(i)) abs(x(i)-x_ref)], a);
       abs(x(i)-x_ref);
       x_s(i)=x(i);
       u(i)=u(i)*sign(theta(i));
      end
    end
  end
  u(i+1)=u(i);
  x_s(i+1)=x_s(i);
  %------------------------

  figure(1)
  plot(t,theta_dot);
  figure(2)
  plot(t,theta*180/pi,'-',t,x*1000,'-.');
  title('Response of the Cart Position and Pendulum Angle');
  xlabel('Time [sec]');
  ylabel('Pendulum Angle [deg], Cart Position [mm]');
  legend('Pendulum Angle', 'Cart Position');
  figure(3)
  plot(t,u);
  title('Control Signal');
  xlabel('Time [sec]');
  ylabel('Input Voltage u(t) [V]');
  figure(4)
  plot(t,theta*180/pi,'-',t,x_s*1000,'-.');
  title('Response of the Cart Position and Pendulum Angle');
  xlabel('Time [sec]');
  ylabel('Pendulum Angle [deg], Cart Position [mm]');
  legend('Pendulum Angle', 'Cart Position');
  figure(5)
  plot(t,theta,'-',t,theta_dot/3,'-.');
  title('Pendulum Angle vs. Pendulum Angular Velocity');
```

```
xlabel('Time [sec]');
ylabel('Pendulum Angle [rad], Angular Velocity*1/3 [rad/sec]');
legend('Pendulum Angle', 'Angular Velocity');
```

Script Title: ode_fuzzy.m

```
%-----------------------------%
% File: ode_fuzzy.m
% defines the system differential equations
% August 2003
% Dilian Stoikov
%-----------------------------%
function dydt=ode_fuzzy(t,y);
global M0 J0 G a first  x_ref stab stab_first x_ref_stab

pend_const %load system physical parameters

x=y(1);
theta=y(2);
d_x=y(3);
d_theta=y(4);
%bc=35;

f3 = ((-J0*(m*lg*d_theta^2*sin(theta)+bc*d_x))-
(m*lg*cos(theta)*(m*g*lg*sin(theta)+bp*d_theta)))/(M0*J0-m^2*lg^2*(cos(theta)^2));
f4 = (1/J0)*(m*lg*cos(theta)*f3-m*g*lg*sin(theta)-bp*d_theta);
g3 = (Ag*Tt/r)*J0/(M0*J0-m^2*lg^2*(cos(theta)^2));
g4 = m*lg*cos(theta)*g3/J0;


if or((180-theta*180/pi)<4, stab==1) %stabilize ?
  stab=1;
  if stab_first==1
    x_ref_stab=x;
    stab_first=0;
  end
    theta=-(pi - theta)*sign(theta);
    xx1=x-x_ref_stab;
    x_n=[xx1, theta, d_x, d_theta]';
    u=-G*x_n;
    theta=  pi + theta;
else %still swinging
  if (abs(d_theta)<0.5)
    d_theta=-sign(theta)*0.2;
  end
  if theta*d_theta<=0
    u=0;
    first=1;
```

```
    else
     if first==1
       first=0;
       x_ref=x;
     end
     u= evalfis([abs(theta*180/pi) abs(d_theta) abs(x-x_ref)], a);%evaluate the contol signal from
the fuzzy model
     abs(x-x_ref);
     u=u*sign(theta);
    end
 end

%calculate second derivatives
dd_x=f3+g3*u;
dd_theta=f4+g4*u;

dydt(1)=d_x;
dydt(2)=d_theta;
dydt(3)=dd_x;
dydt(4)=dd_theta;
dydt=dydt';
```

Script Title: fuzzy_model

```
%-----------------------------%
% File: fuzzy_model.m
% defines the system fuzzy model
% and fuzzy engine
% August 2003
% Dilian Stoikov
%-----------------------------%

% $Revision: 1.2 $
[System]
Name='swing'
Type='sugeno'
NumInputs=3
NumOutputs=1
NumRules=10
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='wtaver'


[Input1]
Name='Pendulum Angle [deg]'
Range=[-180 180]
NumMFs=7
```

```
MF1='S':'gbellmf',[80,20,0]
MF2='PM':'gbellmf',[35 15 117.5]
MF3='PB':'gaussmf',[5 160]
MF4='PVB':'smf',[165 180]
MF5='NM':'gbellmf',[35 15 -117.5]
MF6='NB':'gaussmf',[5 -160]
MF7='NVB':'zmf',[-180 -165]


[Input2]
Name='Angular Velocity [rad/s]'
Range=[-14 14]
NumMFs=5
MF1='S':'gaussmf',[0.9 0]
MF2='PM':'gbellmf',[2,3,6]
MF3='PB':'smf',[7 8]
%MF4='NS':'gaussmf',[0.9 0]
MF4='NM':'gbellmf',[2,3,-6]
MF5='NB':'zmf',[-8 -7]

[Input3]
Name='Cart Position [m]'
Range=[-.18 0.18]
NumMFs=7
MF1='Z':'gaussmf',[0.025 0]
MF2='PS':'gaussmf',[0.02 0.07]
MF3='PM':'gaussmf',[0.012 0.105]
MF4='PB':'smf',[0.13 0.15]
%MF5='NZ':'gaussmf',[0.025 0]
MF6='NS':'gaussmf',[0.02 -0.07]
MF7='NM':'gaussmf',[0.012 -0.105]
MF8='NB':'zmf',[-0.15 -0.13]


[Output1]
Name='Control Signal [V]'
Range=[-8 8]
NumMFs=5
MF1='ZE':'linear',[0   0 0 0]
MF2='PS':'linear',[0   0 0 2]
MF3='PM':'linear',[0   0 0 4]
MF4='PB':'linear',[0   0 0 6]
MF5='NS':'linear',[0   0 0 -2]
MF6='NM':'linear',[0   0 0 -4]
MF7='NB':'linear',[0   0 0 -6]


[Rules]
1 0 1, 4 (1) : 1
1 0 2, 3 (1) : 1
```

```
1 0 3, 2 (1) : 1
1 0 4, 1 (1) : 1
1 2 0, 3 (1) : 1
1 3 0, 2 (1) : 1
2 0 0, 1 (1) : 1
3 1 0, 5 (1) : 1
3 2 0, 1 (1) : 1
4 0 0, 1 (1) : 1
```

D3 Scripts used for the simulations in Chapter 5

Script Title: energy_swing_stab.m

The following script is used for simulations in Chapters 5 and 6. In Chapter 5 only the pendulum swinging is demonstrated and thus the part of the script performs the stabilization simulation is not used. The swing-up and stabilization simulation given in Chapter 6 uses the entire script.

```
%-------------------------------%
%
% File: energy_swing_stab.
% Performs digital simulation
% of the energy based swing controller
% and the linear stabilization controller
% August 2003
% Dilian Stoikov
%-------------------------------%

pend_const; %load physical system parameters
```

```
T = 1/500; %sampling period
Time = 12.5 %simulation time in sec

%--------------
%flags
t_max = 120*pi/180;
Dis = 0.08
movingDown = 0;
stab = 0; %flag for stabilization entering
first = 1;
waitFlag=0;
ePrev = 0;
ePrev_s = 0;
first=1;
%bc=35.4
%bb=2.4;
%----------------

M0 = M+m;
J0 = Jp+m*lg^2;

%--------------------------%
%find the Jacobian necessary for computing the linearized model

x_1 = sym('x1','real');
x_2 = sym('x2','real');
x_3 = sym('x3','real');
x_4 = sym('x4','real');
u_s = sym('u','real');

f_3 = ((-J0*(m*lg*x_4^2*sin(x_2)+bc*x_3))-
(m*lg*cos(x_2)*(m*g*lg*sin(x_2)+bp*x_4)))/(M0*J0-m^2*lg^2*(cos(x_2)^2));
f_4 = (1/J0)*(m*lg*cos(x_2)*f_3-m*g*lg*sin(x_2)-bp*x_4);
g_3=( Ag*Tt/r)*J0/(M0*J0-m^2*lg^2*(cos(x_2)^2));
g_4=-m*lg*cos(x_2)*g_3/J0;
f_s=[x_3, x_4, f_3, f_4]';
%F_jacobian = jacobian(f_s,[x_1 x_2 x_3 x_4])
f_s1=[x_3, x_4, f_3+g_3*u_s, f_4+g_4*u_s]';
F_s=jacobian(f_s1,[x_1 x_2 x_3 x_4]);
G_s=jacobian(f_s1,[u_s]);
%-------------------------%


%-------------------------
%stabilizing linear state-space controller derivation
A=subs(F_s,[x_2,x_3,x_4,u_s],[0,0,0,0]);
B=subs(G_s,x_2,0);
C = [1 0 0 0; ...
     0 1 0 0];
```

```
Q_opt = diag([3e6 7*180/pi 0.1 1]);
Q_opt = diag([3e6 150 1 1200]);
optR = 1000;
optR=2000;
Gtarget = lqr(A,B,Q_opt,optR);
%poles =  [-6.04 -7.27 -12.05 -13.05];
poles = [-5.2-j*3, -5.2+j*3, -3+j*3.8, -3-j*3.8];
%poles=eig(A-B*Gtarget)
%equivalent discrete system
[Phi, Gamma] = c2d(A,B,T);
G = place(Phi, Gamma, exp(poles*T));



t=0:T:Time;
l = length(t);
u=zeros(1,l);
u_s=zeros(1,l);
x=zeros(4,l);
x_s=zeros(4,l);
pos=0;
Dis=0;
%set initial condition for system
x(:,1) = [0, 40*pi/180,0, 0]';
x(:,2) = [0,  40*pi/180,0, 0]';
x_s(:,1) = [0, 40*pi/180,0, 0]';
x_s(:,2) = [0, 40*pi/180,0, 0]';



x_ref=x(1,1);
x_ref_s=x_s(1,1);
for i= 2:l-1
  %-----------------------------------
  linear stabilizing controller
  if or(and((180-abs(x(2,i))*180/pi)<2.5,abs(x(4,i))<1), stab==1)% check whether stabiliztion
                                                                  %citeria are met
     stab=1;
     if first==1
       x(1,i)=x(1,i-1);
       x_ref_stab=x(1,i);
       first=0;
     end
     x(2,i)=-(pi - abs(x(2,i)))*sign(x(2,i));
     xx1=x(1,i)-x_ref_stab;
     x(1,i)=x(1,i)-x_ref_stab;
     x_n=[xx1, x(2,i), x(3,i), x(4,i)]';
     u(i)=-K*x_n;
     %check safety range for the voltage across motor
     if abs(Ra*Ag*u(i)+(E*x(3,i)/r1))>27
```

```
      fprintf('Voltage across motor exceeds safety region');
    end
    x(:,i+1)=Phi*x_n+Gamma*u(i);% retrieve next system state
    %make observations
    Z0(:,i+1)=F0*Z0(:,i)+H0*u(i)+G0*[x(1,i+1);x(2,i+1)]+Li*[x(1,i+1)-x(1,i); x(2,i+1)-x(2,i)];
    x(3,i+1)=Z0(1,i+1);
    x(4,i+1)=Z0(2,i+1);
    x(1,i)=x(1,i)-x_ref_stab;
    x(1,i+1)=x_ref_stab+x(1,i+1);
    x_s(1,i+1)=x(1,i+1);
    x(2,i+1)= pi + x(2,i+1);
    x(2,i)=   pi + x(2,i);
%----------------------------------%
else % swing
    thetaPrev = x(2,i-1);
    theta = x(2,i);
    absTheta = abs(theta);
    u(i)=0;
    if movingDown == 1
      if and(absTheta<t_max, theta*sign(thetaMax)>=0) %Cart moving,
        pos = sign(thetaMax)*(Dis+Dis*cos(theta/thetaMax*pi));
        eCurrent = (x_ref + pos) - x(1,i);
        eCurrent_s = (x_ref_s+pos) - x_s(1,i);
        u(i) = 0.87937838657944*u(i-1) + 590.610000000000*eCurrent -
             582.717808348398*ePrev; % + 0.35*sign(thetaMax);
       %u(i) = (0.94698287025778*u(i-1) + 559.045036917509*eCurrent –
              539.327178526916*ePrev);
        %check safety range for the voltage across motor
        if abs(Ra*Ag*u(i)+(E*x(3,i)/r1))>27
          fprintf('Voltage across motor exceeds safety region');
        end
        u_s(i) = 0.87937838657944*u_s(i-1) + 590.610000000000*eCurrent_s –
             582.717808348398*ePrev_s;  % + 0.35*sign(thetaMax);
        %u_s(i) = (0.94698287025778*u_s(i-1) + 559.045036917509*eCurrent_s –
              539.327178526916*ePrev_s);
        ePrev = eCurrent;
        ePrev_s = eCurrent_s;
      else  %cart's not moving
        if theta*sign(thetaMax)<0
         movingDown =0;
        end
        u(i) = 0;
        x(1,i) = x(1,i-1);
        x_s(1,i) = x_s(1,i-1);
      end
    end
    if movingDown == 0  % moving up
      u(i) = 0;
      x_s(1,i) = x_s(1,i-1);
      if abs(thetaPrev) > absTheta%starts moving down
        thetaMax = thetaPrev;
```

```
    Dis=displ(abs(thetaMax));%calculate cart's excursion
    x_ref=x(1,i);
    x_ref_s=x_s(1,i);
    movingDown = 1;
    start_down_first=1;
    ePrev = 0;
    ePrev_s = 0;
    if abs(thetaMax) > t_max
      thetaMax = sign(thetaMax)*t_max;
    end
  end
end


%-----------------------------------------------%
%retrive linearized model of the
%system about the current system state and calculate the next state
%
f3=((-J0*(m*lg*x(4,i)^2*sin(x(2,i))+bc*x(3,i)))-
        (m*lg*cos(x(2,i))*(m*g*lg*sin(x(2,i))+bp*x(4,i))))/
                        (M0*J0-m^2*lg^2*(cos(x(2,i))^2));

f=[x(3,i); x(4,i); f3; (1/J0)*(m*lg*cos(x(2,i))*f3-m*g*lg*sin(x(2,i))-bp*x(4,i))];

% the numerical form of the Jacobian is constructed from the symbolic one for faster
% calculations
delta_f= [0 0 1 0;
          0 0 0 1;
          0, (-        34860471495937659/36028797018963968000*x(4,i)^2
              *cos(x(2,i))+27/500*sin(x(2,i))*(26487/50000*sin(x(2,i))+1/1000*x(4,i))-

                    715149/25000000*cos(x(2,i))^2)/(5284828984715659/2882303761517
                    11744-1680959553716783/576460752303423488*cos(x(2,i))^2)-
                    1680959553716783/288230376151711744*(-
                    34860471495937659/36028797018963968000*x(4,i)^2*sin(x(2,i))-
                    228529757584480209/360287970189639680*x(3,i)-
                    27/500*cos(x(2,i))*(26487/50000*sin(x(2,i))+1/1000*x(4,i)))/(528482
                    8984715659/288230376151711744-
                    1680959553716783/576460752303423488*cos(x(2,i))^2)^2*cos(x(2,i)
                    )*sin(x(2,i)), -
                    228529757584480209/360287970189639680/(5284828984715659/288
                    230376151711744-
                    1680959553716783/576460752303423488*cos(x(2,i))^2), (-
                    34860471495937659/18014398509481984000*x(4,i)*sin(x(2,i))-
                    27/500000*cos(x(2,i)))/(5284828984715659/288230376151711744-
                    1680959553716783/576460752303423488*cos(x(2,i))^2);
                    0, -106036120328902371/35184372088832000*sin(x(2,i))*(-
                    34860471495937659/36028797018963968000*x(4,i)^2*sin(x(2,i))-
                    228529757584480209/360287970189639680*x(3,i)-
                    27/500*cos(x(2,i))*(26487/50000*sin(x(2,i))+1/1000*x(4,i)))/(528482
                    8984715659/288230376151711744-
```

```
                    1680959553716783/576460752303423488*cos(x(2,i))^2)+106036120
                    328902371/35184372088832000*cos(x(2,i))*(-
                    3486047149593765 9/3602879701896396 8000*x(4,i)^2*cos(x(2,i))+27
                    /500*sin(x(2,i))*(26487/50000*sin(x(2,i))+1/1000*x(4,i))-
                    715149/25000000*cos(x(2,i))^2)/(5284828984715659/2882303761517
                    11744-1680959553716783/576460752303423488*cos(x(2,i))^2)-
                    17824242950593083097451179119 2493/10141204801825835211973 6
                    25643008000*cos(x(2,i))^2*(-
                    3486047149593765 9/3602879701896396 8000*x(4,i)^2*sin(x(2,i))-
                    228529757584480209/3602879701896396 80*x(3,i)-
                    27/500*cos(x(2,i))*(26487/50000*sin(x(2,i))+1/1000*x(4,i)))/(528482
                    8984715659/288230376151711744-
                    1680959553716783/576460752303423488*cos(x(2,i))^2)^2*sin(x(2,i))
                    -1040214340426532259 51/3518437208883200000*cos(x(2,i)), -
                    2423240887396283269273985354267 5539/1267650600228229401496
                    7032053760000*cos(x(2,i))/(5284828984715659/28823037615171174
                    4-1680959553716783/576460752303423488*cos(x(2,i))^2),
                    106036120328902371/35184372088832000*cos(x(2,i))*(-
                    3486047149593765 9/1801439850948198 4000*x(4,i)*sin(x(2,i))-
                    27/500000*cos(x(2,i)))/(5284828984715659/288230376151711744-
                    1680959553716783/576460752303423488*cos(x(2,i))^2)-
                    3927263715885273/70368744177664000];

      g3=( Ag*Tt/r)*J0/(M0*J0-m^2*lg^2*(cos(x(2,i))^2));
      G_u=[0; 0; g3; m*lg*cos(x(2,i))*g3/J0];

      ai=zeros(4,4);
      for icount=1:4
        fd=delta_f(icount,:)'+ (f(icount,:)-x(:,i)'*delta_f(icount,:)')*x(:,i)/(norm(x(:,i),2)^2);
        ai(icount,:)=fd';
      end

      A_nonLinear=[ai(1,:); ai(2,:); ai(3,:); ai(4,:)];
      [Phi_nonLinear, Gamma_nonLinear] = c2d(A_nonLinear, G_u,T);
      x(:,i+1) = Phi_nonLinear*x(:,i) + Gamma_nonLinear*u(i);
      x_s(:,i+1) = Phi_nonLinear*x_s(:,i) + Gamma_nonLinear*u_s(i);
      %--------------------
    end %if
  end %for


  u(i+1)=u(i);

  figure(1)
  plot(t,x(2,:)*180/pi,'-',t,1000*x_s(1,:),'-.');
  title('Response of the Cart Position and Pendulum Angle');
  xlabel('Time [sec]');
  ylabel('Pendulum Angle [deg], Cart Position [mm]');
  legend('Pendulum Angle', 'Cart Position');
  figure(2)
  plot(t,x(2,:)*180/pi),'-',t,x(4,:)/3,'-.');
```

```
title('Pendulum Angle vs. Pendulum Angular Velocity');
xlabel('Time [sec]');
ylabel('Pendulum Angle [rad], Angular Velocity*1/3 [rad/sec]');
legend('Pendulum Angle', 'Angular Velocity');
figure(3)
plot(t,u);
title('Control Signal');
xlabel('Time [sec]');
ylabel('Input Voltage u(t) [V]');
```

Script Title: cart_trajectory.m

```
%-----------------------------------%
% File: cart_trajectory.m
% Simulates the movement of the cart
% under the pendulum angle driven
% trajectory
% August 2003
% Dilian Stoikov
%-----------------------------------%
global   Amp thetaMax movingDown

time=0.5;
thetaMax = 90*pi/180;%Angle at which cart motion begins %motion ends at theta=0
Amp=0.1;%cart motion amplitude
moving_Down = 0;

y0=[100*pi/180 0]; %initial angle and velocity of pendulum
[t y]=ode45(@ode_theta,[0 time], y0); %use ODE solver for theta and theta_dot
theta=y(:,1);
theta_dot = y(:,2);
theta_deg=theta*180/pi;

%calculate cart position x_M and velocity dx_M from theta
n=size(t);
x_M=zeros(n);
dx_M=zeros(n);
for i=1:n
  if and(theta(i)<=thetaMax, theta(i)>0)
    x_M(i)=Amp*cos(theta(i)*pi/thetaMax);
    dx_M(i)=-Amp/thetaMax*sin(theta(i)/thetaMax*pi)*theta_dot(i);
  elseif theta(i)>thetaMax
     x_M(i)=-Amp;
     dx_M(i)=0;
  else
     x_M(i)=Amp;
     dx_M(i)=0;
  end
```

```
end

figure(1)
plot(theta_deg,1000*x_M)
title('Cart Motion with Prescribed Trajectory');
xlabel('Pendulum Angle (deg)');
ylabel('Cart Position (mm)');
figure(2);
plot(t,theta,'-',t,dx_M,'-.');
title('Cart Velocity');
xlabel('Time (sec)');
ylabel('Pendulum Angle (rad), Cart Velocity (m/s)');
```

Script Title:ode_theta.m

```
%-----------------------------------------------%
%File: ode_theta.m
% ode solver function used by
% cart_trajectory.m for theta and theta_dot
%
% August 2003
% Dilian Stoikov
%-----------------------------------------------%
function dydt=ode_theta(t,y);
  global  Amp thetaMax movingDown
  m=0.2140;
  Jp=0.0043;
  bp=0.001;
  g=9.8;
  lg=.25;

  theta=y(1);
  d_theta=y(2);

  abstheta = abs(theta);
  if and(abstheta<=thetaMax, theta>=0) %Cart moving, dd_theta involves cart acceleration
     coef1=-(Amp*pi^2/thetaMax^2)*cos(theta/thetaMax*pi)*d_theta^2;
     coef2=-(Amp*pi/thetaMax)*sin(theta/thetaMax*pi);
     dd_theta=(m*lg*cos(theta)*coef1-m*g*lg*sin(theta)-bp*d_theta)/(m*lg^2+Jp-
          m*lg*coef2*cos(theta));
  else  % cart's not moving dd_x = 0;
       dd_theta=(-m*g*lg*sin(theta)-bp*d_theta)/(m*lg^2+Jp);
  end

 dydt(1)=d_theta;
 dydt(2)=dd_theta;
 dydt=dydt';
 thetaPrev = theta;
```

Appendix E
Controller C Source Code

Server Application

```
/********************
 * pendulum_control.c
**********************/
#include "global.h"
#include "enc.h"
#include "eserrlp.h"
#include "msg.h"

/*Global variable initialization */
int Board_Base_Address = 0x220;
int Read_Port_Address = 0x203;
struct encReference initValues;
long double outputArray[dataLength][dataWidth];
long double sampleRateHz=500;
long double samplePeriod = (long double) (1.0/sampleRateHz);
unsigned count = 0;
unsigned timeUpdate;
unsigned timeConst;
int l;
FILE *f_in, *fshake, *fstand;



int main(void) {
   pthread_t  serverThreadId;
   pthread_attr_t controlThrAtrs;
   sigset_t signalSet;

   atexit(finish);
   ThreadCtl (_NTO_TCTL_IO, NULL);
   sigfillset ( &signalSet );
   /*
   * Block signals in initial thread. New threads will
   * inherit this signal mask.
   */
   pthread_sigmask ( SIG_BLOCK, &signalSet, NULL );
   /*Initialize controller hardware*/
   initHardware();
   // open files for data storing
   if ((f_in=fopen("fin.txt", "w")) == NULL) {
    fprintf (stderr, "Unable to open file\n");
    exit (EXIT_FAILURE);
   }
   if ((fshake=fopen("shake.txt", "w")) == NULL) {
    fprintf (stderr, "Unable to open file\n");
    exit (EXIT_FAILURE);
```

```
    }
    if ((fstand=fopen("stand.txt", "w")) == NULL) {
      fprintf (stderr, "Unable to open file\n");
      exit (EXIT_FAILURE);
    }
    //create the server thread to receive msgs from the client
    pthread_create(&serverThreadId, NULL, serverThread, NULL);
    pthread_join(serverThreadId, NULL);
    DAC_OUT(0.0);
    return(EXIT_SUCCESS);

  }


  void finish(void) {
    DAC_OUT(0.0);
    //perform additional cleaning if necessary
  }



  /************************
   file serverThead
  ************************/
  #include "msg.h"
  #include "global.h"

  timer_t  timerId;
  struct sigevent event;
  struct itimerspec timerSpec;
  pthread_t controlThreadId;
  pthread_attr_t controlThrAtrs;
  pthread_mutex_t mutex =  PTHREAD_MUTEX_INITIALIZER; //mutex used for sync. access to
                                                      //feedback gain coefs.
  long double G_new[4]; //for gain coefs sent by client
  FILE *pidFp;
  const int sSize=512;
  char msgString[sSize];
  MESSAGE rmsg;
  int channelID;
  struct _msg_info info;

  int msgParse (const MESSAGE *msg, char* result,
            struct _msg_info *info);
  void createControlThread(void);
  void* serverThread(void* arg) {

    /* Create a channel for client connections */
    channelID = ChannelCreate (0);
    /* Advertise this logging service by writing to a .pid file */
    if ((pidFp=fopen("/home/Channels/channel.pid", "w")) == NULL) {
      fprintf (stderr, "Unable to open file\n");
```

```
    exit (EXIT_FAILURE);
  }
  fprintf (pidFp, "0 %d %d\n", getpid(), channelID);
  fclose (pidFp);
  //create mutex for sync access to new gain coefs
  if(pthread_mutex_init(&mutex,NULL) != EOK) {
    printf("can't init mutex");
    exit(EXIT_FAILURE);
  }

  //initilize the G_new with the values corresponding to the default poles
  G_new[0] = -3.526909355076609e+1;
  G_new[1] = -4.081770631201547e+1;
  G_new[2] = -1.685871061154501e+1;
  G_new[3] = -7.39931708588781e+0;

  /* Main message receiving loop */
  while (1) {
    /* Get message, parse it and reply ASAP */
    int clientID = MsgReceive (channelID, &rmsg, sizeof(MESSAGE), &info);
    int status = msgParse (&rmsg, msgString, &info);
    MsgReply (clientID, status, NULL, 0);
    /* Log the message and flush for immediate update */
    printf ("%s", msgString);     fflush (stdout);
    if (status == MSG_END) break;
  }
}

/* Parse the message and take appropriate action based on the header */
int msgParse (const MESSAGE *msg, char* result,
        struct _msg_info *info) {
  int i;
  switch (msg->m_hdr) {
   case MSG_DATA:
     sprintf (result, "%d/%d/%d sent: %Lf %Lf %Lf,%Lf\n",
         info->nd, info->pid, info->chid, msg->m_data[0], msg->m_data[1],msg->m_data[2],msg-
                                                                                  >m_data[3]);
     pthread_mutex_lock(&mutex);
     for(i=0;i<4;i++) {
        G_new[i] = msg->m_data[i];
     }
     pthread_mutex_unlock(&mutex);
     return MSG_OK;
    case MSG_START:
     //create control thread
     createControlThread();
     return MSG_OK;
    case MSG_END:
     sprintf (result, "Shutdown message received\n");
     return MSG_END;
    default:
     sprintf (result, "Unknown message type %d received."
         " Ignoring.\n",msg->m_hdr);
```

```
    return MSG_INVALID;
  }
}


/*************************
 function controlThread
 creates the control thread and timer
***************************/
void createControlThread(void) {
  pthread_attr_init(&controlThrAtrs); //init the thread's atribute strcture
  pthread_attr_setinheritsched(&controlThrAtrs, PTHREAD_EXPLICIT_SCHED);
  controlThrAtrs.param.sched_priority = 63; //highest priority
  pthread_attr_setschedpolicy(&controlThrAtrs, SCHED_FIFO);
  pthread_create(&controlThreadId, &controlThrAtrs, controlThread, NULL);
  //Create the timer
  SIGEV_SIGNAL_INIT( &event, SIGALRM );  //initialize the event structute for sending a signal
  if (timer_create(CLOCK_REALTIME, &event, &timerId) == -1) {
    fprintf (stderr, "can't cretae timer, erno 5\n");
    perror(NULL);
    exit (EXIT_FAILURE);
  }
  //set the timer 2 ms delay 2ms reload -> 500Hz
  timerSpec.it_value.tv_sec = 0;
  timerSpec.it_value.tv_nsec = samplePeriod*1e9;
  timerSpec.it_interval.tv_sec = 0;
  timerSpec.it_interval.tv_nsec = samplePeriod*1e9;
  //start the timer
  timer_settime(timerId,  0, &timerSpec, NULL);

}



/*************************
* File: controlThread.c
* Provides routines for swing up
* and stabilization controllers
*************************/
#include "global.h"
#include "enc.h"
#include "eserrlp.h"

unsigned int shakeLeft = 0;
unsigned int stabilezeFlag = 0;
unsigned int standUpFlag = 0;
unsigned int standUpFirst = 1;
unsigned int recoveryFlag = 0;
unsigned int moveFlag = 1;
unsigned int cycleCount = 1;
```

```
int flag = 1; /*controller failure flag*/
float outData[3000];
int angle, cart, volt;
int h;
long double U = 0.0;
long double dacVolt = 9.8;
int vFlag = 0;
long double xCurrent;
int upCounter = -1;


long int absRef;   //to keep the starting point of the cart's moving

/*Controller Constants */
long double G[4] = { 0, 0, 0, 0 };  /* Feedback gains*/
long double Ao[2][2], Bo[2], Co[2][2], K[2][2]; /*Integral observer const*/



//functions prototypes
void loadPendulumControllerConstants(void);
void pendulumContorllerIntgObsv(void);
void positionController(long double xRef, unsigned int clearFlag);
void shake(void);
void standUpRoutine(void);
void selectAmplitude(float th, float* amp);


void* controlThread(void* arg) {

  sigset_t localSet;
  int sig,i;
  long double theta;
  sigfillset ( &localSet );

  loadPendulumControllerConstants();  //initialize controller constants
  for(;;) {
    sigwait(&localSet, &sig);  // wait for a signal
    if(sig == SIGINT) {
      exit(EXIT_SUCCESS);
    }
    if(sig != SIGALRM) {      // check whether signal is from timer
      continue;
    }
    if(shakeFlag) {
      shake();
    }
    if(standUpFlag) {
      standUpRoutine();
    }
    if(stabilezeFlag) {
      if(pthread_mutex_trylock(&mutex) == EOK) {
        for(i=0;i<4;i++) {
```

```
            G[i] = G_new[i];
          }
        pthread_mutex_unlock(&mutex);
        }
        pendulumContorllerIntgObsv();
      }
    }
  }


/***********************************
  function loadPendulumControllerConstants
************************************/

void loadPendulumControllerConstants(void) {

  //pend1 @500Hz
  G[0] = -3.526909355076609e+1;
  G[1] = -4.081770631201547e+1;
  G[2] = -1.685871061154501e+1;
  G[3] = -7.39931708588781e+0;

  Ao[0][0] = 9.6000000000000e-1;
  Ao[0][1] = 0.0000000000000e+1;
  Ao[1][0] = 0.0000000000000e+1;
  Ao[1][1] = 9.5000000000000e-1;

  Bo[0] =  1.278244678708e-2;
  Bo[1] =  -3.852281524850e-2;

  Co[0][0] = 0.000000000000e+0;
  Co[0][1] = -3.71037748177e-3;
  Co[1][0] = 0.000000000000e+0;
  Co[1][1] = 7.031140722733e-2;

  K[0][0] = 1.721110251918242e+1;
  K[0][1] = 3.50207411350e-3;
  K[1][0] = 8.40497787240069e+0;
  K[1][1] = 2.493363592778785e+1;
}


/***********************************
  function pendulumControllerIntgObsv
  Performs stabilization of the inverted pendulum
  Initially written by Ryan Mowat
  Modified by Dilian Stoikov
************************************/
void pendulumContorllerIntgObsv(void) {

  static long double X[4] = { 0.0, 0.0, 0.0, 0.0 };
  static long double Xold[4] = { 0.0, 0.0, 0.0, 0.0 };
```

```
    static int outputcount;

    //read encoders to find position
    //cart position (2*Pi/(4*1024)*r)*encoder_reading
    X[0] = ((long double) (-readCntr(axisA, ENC_baseAddress) + initValues.aRef))*0.0000389631120;
    //pendulum position = (2*Pi/(4*512))*encoder_reading
    X[1] = ((long double) (readCntr(axisB, ENC_baseAddress) - initValues.bRef))*0.0030679615757;
    X[1] = -((PI - fabs(X[1]))*SGN(X[1]));
    //Make integral observer calculations
    X[2] = Ao[0][0]*Xold[2] + Ao[0][1]*Xold[3] + Bo[0]*U + Co[0][0]*X[0] + Co[0][1]*X[1] +
K[0][0]*(X[0]-Xold[0]) + K[0][1]*(X[1] - Xold[1]);
    X[3] = Ao[1][0]*Xold[2] + Ao[1][1]*Xold[3] + Bo[1]*U + Co[1][0]*X[0] + Co[1][1]*X[1] +
K[1][0]*(X[0]-Xold[0]) + K[1][1]*(X[1] - Xold[1]);
    //Calculate control output
    U = -(G[0]*X[0] + G[1]*X[1] + G[2]*X[2] + G[3]*X[3]);//+0.75*SGN(X[2]);
    //check security range
    if( (fabs(X[1])*180/PI > angleLimit) || (fabs(X[0]) > cartLimit) || (fabs(U) >
(percentOver*currentLimit)) ) {
    if(fabs(X[1])*180/PI > angleLimit) { angle=1;}
    if(fabs(X[0]) > cartLimit)  {cart = 1; }
    if(fabs(U) >= (percentOver*currentLimit)) {volt = 1;}      //
        flag = 0;
    }
    if(flag) {
      DAC_OUT(U);
    }
    else {
      U=0.0;
      DAC_OUT(U);
    }
    Xold[0] = X[0];
    Xold[1] = X[1];
    Xold[2] = X[2];
    Xold[3] = X[3];
}

/*****************************
 function positionCotroller
 Implements digital lead compensator
*******************************/
void positionController(long double xRef, unsigned int clearFlag) {

    long double eCurrent;
    static long double ePrev = 0.0;
    static long unsigned int outputcount = 0;
    if(clearFlag) {
      U=0.0;
      ePrev = 0.0;
      DAC_OUT(0.0);
      return;
    }

    //find current position of cart
```

```
  xCurrent = ((long double) (-readCntr(axisA, ENC_baseAddress) +
initValues.aRef))*0.0000389631120;
  eCurrent = xRef - xCurrent;
  //calculate voltage
  U = (0.87937838657944*U + 590.61*eCurrent - 582.717808348398*ePrev);// +
0.15*SGN(eCurrent);
  ePrev = eCurrent;
  if(U>=10.0) {
    DAC_OUT(9.8);
     vFlag = 1;
  }
  else {
    DAC_OUT(U);
  }
}


/**************************
 function shake
 nitial pendulum exciting
**************************/
void shake (void) {

  static int loopCountRight = 200;
  static  int loopCountLeft = 200;

  if(!shakeLeft) {
   if (loopCountRight > -1) {
     positionController(-0.07*(1+cos(loopCountRight/200.0*PI)),0);
    loopCountRight--;
   }
   else {
    positionController(1.0,1);
      if(((long double) (readCntr(axisB, ENC_baseAddress) - initValues.bRef))*0.0030679615757 >
0);//wait to pass down position
      shakeLeft = 1;
     initValues.aRef = readCntr(axisA, ENC_baseAddress);
    }

  }

  else {
    if (loopCountLeft > -1) {
     positionController(0.07*(1+cos(loopCountLeft/200.0*PI)),0);
     loopCountLeft--;
    }
    else {
     positionController(-1.0,1);
     shakeFlag = 0;
     standUpFlag = 1;
      absRef = readCntr(axisA, ENC_baseAddress);  // save the current position of the cart for
reference
    }
```

```
    }
  }

  /************************************
   function standUpRoutine
   defines cart trajectory
   based on pendulum driven constraint
   equation
  *************************************/
  void standUpRoutine(void) {

    static unsigned int movingDown = 0;
    static unsigned int waitFlag = 0;
    static int direction;
    static unsigned int recCount;
    long double thetaCurrent,absThetaCurrent, absThetaMax;
    static long double thetaPrev = 0.0;
    static long double thetaMax,dTheta_dt0;
    static float A = 0.0;
    static unsigned int outputcount;
    float cosValue;
    long double angle;

    //find the position of the pendulum
    thetaCurrent = ((long double) (readCntr(axisB, ENC_baseAddress) -
initValues.bRef))*0.0030679615757;
    absThetaCurrent = (long double) (fabs(thetaCurrent));
    if( (PI - fabs(thetaCurrent)) < 1.5*PI/180 )  { //stabilization condition reached ?
      initValues.aRef = readCntr(axisA, ENC_baseAddress);
      stabilezeFlag = 1;
      standUpFlag = 0;
      recCount = 0;
      return;
    }
    if(movingDown) {
      if(waitFlag) {
       if(absThetaCurrent <= 110 * PI / 180) { //wait to swing below threshold
         waitFlag = 0;
       }
       else {
        return;
       }
       }
      if((thetaCurrent*direction) >= 0.0 ) {
        if(moveFlag) {
           cosValue = 1+ cos((double) (2*PI - thetaCurrent / thetaMax * PI));
                positionController((direction*A*cosValue), 0);
        }
      }
      else {       //stop moving
         positionController(1.0, 1);
         movingDown = 0;
      }
```

```
    }
    else { //swinging up
      if( absThetaCurrent < fabs(thetaPrev) ) { //start swinging down?
        thetaMax = thetaPrev;
        absThetaMax = fabs(thetaMax);
        if(recoveryFlag) {
          if(absThetaMax > 70.0 * PI / 180.0) {   //wait until swinging below 70 deg
                moveFlag = 0;
          }
          else {
            if(SGN(thetaMax) == 1.0) {    //see whether move down from left -> go right
                  recoveryFlag = 0;
                  moveFlag = 1;
            }
          }
        }
        if (absThetaMax > 140.0 * PI / 180.0 && absThetaMax < 180.0 * PI / 180.0) {
          recCount++;
          if(recCount > 14) {        //do 14 swings before start again
                recCount = 0;
                recoveryFlag = 1;
                positionController(1.0,1);
                        moveFlag = 0;
                }
              }
              movingDown = 1;
              direction = SGN(thetaMax);
              selectAmplitude(thetaMax, &A);
              if(fabs(thetaMax) > 120.0 * PI / 180.0) {
                thetaMax = direction*120.0 * PI / 180.0;
                waitFlag = 1;
              }
              initValues.aRef = readCntr(axisA, ENC_baseAddress);
        }
    }
    thetaPrev = thetaCurrent;

    //write data into files
    fprintf(f_in,"%Lf\n",thetaCurrent*180/PI);
    fprintf(fshake,"%Lf\n",((long double) (-readCntr(axisA, ENC_baseAddress) +
absRef))*0.0000389631120);
    fprintf(fstand,"%Lf\n",U);
}
```

```
/*******************************
 file inti.c
 hardware setup
*******************************/
#include "global.h"
#include "enc.h"
#include "eserrlp.h"

void initHardware(void) {
   unsigned uError;
   //initialize ADC
   uError = Setup_PNP_Board();
   if (uError) {return;}
   //Configure the ADC
   Board_Write(AO_Configuration_Register,0x0001);
   AO_Reset_All();
   AO_Board_Personalize();
   AO_LDAC_Source_And_Update_Mode();
   DAC_OUT(0.0);
   initEncoder(axisA,MCR,ICR,OCCR,QR,ENC_baseAddress);
   initEncoder(axisB,MCR,ICR,OCCR,QR,ENC_baseAddress);
   /*Load preset encoder value to counters*/
   loadCntr(wrtAllAxis, ENC_Preset, ENC_baseAddress);
   /*establish reference position */
   initValues.aRef = readCntr(axisA,ENC_baseAddress);
   initValues.bRef = readCntr(axisB,ENC_baseAddress);
}


/*******************************
 File encfun.c
 provides functions for communication
 with the optical encoder borad
 Written by Ryan Mowat 1995
 Modified by Dilian Stoikov, Aug 2003
********************************/
#include "global.h"
#include "enc.h"
#include "eserrlp.h"

/*  TILE: initEncoder
  * Initialize encoder counters
  * ip - indirect address(port) of axis
  * mcr - Master Control Registry init value
  * icr - Input Control registry init value
  * occr - Output/Counter Control Registry init value
  * qr - Quadrature Register init value */
void initEncoder(int ip, int mcr, int icr, int occr, int qr, short bAddr) {
  writeIp(ip+1, mcr, bAddr);
  writeIp(ip+1,icr,bAddr);
  writeIp(ip+1,occr,bAddr);
```

```
  writeIp(ip+1, qr, bAddr);
}

/*  TITLE: loadCNTR
     * Breaks the supplied number into 3 bytes and writes them
     * to the specified encoder */
void loadCntr(int ip, int num, short bAddr) {
  loadPr(ip, num, bAddr);  /*load preset register*/
  writeIp(ip+1, PR_CNT, bAddr); /*transfer to counter*/
}


/*  TITLE: loadPr
     * Loads the specified preset register of the specified encoder
     * entries: ip - indirect port of axis
     * num - number to be loaded */
void loadPr(int ip, int num, short bAddr) {
     tByte temp;                /*conversion variable */
     temp.n = num;               /*convert */
     writeIp(ip+1, RST_ADDR, bAddr);  /* reset Pr address */
     writeIp(ip, temp.tb.lsb, bAddr); /*send bytes */
     writeIp(ip, temp.tb.lsb_1,bAddr);
     writeIp(ip, (0x00),bAddr);
}

/*  TITLE: readCntr
  *Reads the counter of the specified encoder */
int readCntr(int ip, short bAddr) {
     int num;
     writeIp(ip+1,CNT_OL,bAddr);   /*dump counter to output latch */
     num = readOl(ip,bAddr);      /*read output latch*/
     return(num);
}


/*  TITLE  readOl
     *readsoutput latch of the specified encoder */
int readOl(int ip, short bAddr) {
     tByte temp;
     int num;
     temp.n  = (0x0000);
     writeIp(ip+1, RST_ADDR, bAddr);
     temp.tb.lsb = readIp(ip, bAddr);
     temp.tb.lsb_1 = readIp(ip,bAddr);
     temp.tb.msb = readIp(ip, bAddr);
     num = temp.n;                /*convert*/

     return(num);

}

/*  TITLE: writeIp
     * writes a byte to a specified indirect port */
```

```
void writeIp(int ip, int val, short bAddr) {
    out8(bAddr, (unsigned char) ip);
    out8(bAddr+1, (unsigned char) val);
}

/*  TITLE: readIp
    * reads a byte from the specified indirect port */

int readIp(int ip, short bAddr) {
    int val = 0;
    out8(bAddr, (unsigned char) ip);
    val = in8(bAddr+1);
    return(val);

}



/****************************************
* This file is a collection of register level E-Series functions
* Included in this file are:
* DAQ_STC_Windowed_Mode_Write
* DAQ_STC_Windowed_Mode_Read
* Board_Write
* Board_Read
  Functions provided by National Instruments
  Modified for work under QNX Neutrino by Dilian Stoikov
****************************************/


#include "global.h"
#include "enc.h"
#include "eserrlp.h"
#include <hw/inout.h>



/* This function implements the STC windowed writing mode.  First *
 * the register address is written to the Window_Address_Register.*
 * The data is then written to the Window_Data_Write_Register.   *
 * Both writes are unsigned 16 bit.                              */

void DAQ_STC_Windowed_Mode_Write( unsigned uRegister_Address,  unsigned uData)
{

  out16(Board_Base_Address + Window_Address_Register * 2,uRegister_Address);
  out16(Board_Base_Address + Window_Data_Write_Register * 2,uData);
}

/* This function implements the STC windowed reading mode.  First *
 * the register address is written to the Window_Address_Register.*
 * The data is then read from the Window_Data_Write_Register.    *
```

```
 * Both the read and write are unsigned 16 bit.              */

unsigned DAQ_STC_Windowed_Mode_Read(unsigned uRegister_Address)
{
   out16(Board_Base_Address + Window_Address_Register * 2,uRegister_Address);
   return(in8(Board_Base_Address + Window_Data_Read_Register * 2));
}

/* This function writes to an E-Series board register          */

void Board_Write(unsigned uRegister_Address, unsigned uData)
{
   out16(Board_Base_Address + uRegister_Address,uData);
}

/* This function reads from an E-Series board register          */

unsigned Board_Read(unsigned uRegister_Address)
{
   return(in16(Board_Base_Address + uRegister_Address));
}
 /*
 * Use this function to reset the DAQ-STC.
 */

void AO_Reset_All()
{

 /*
 * Writing to register Joint_Reset_Register with address 72.
 *    AO_Configuration_Start <= 1
 * New pattern = 0x0020
 */
 DAQ_STC_Windowed_Mode_Write(Joint_Reset_Register,0x0020);

 /*
 * Writing to register AO_Command_1_Register with address 9.
 *    AO_Disarm <= 1
 * New pattern = 0x2000
 */
 DAQ_STC_Windowed_Mode_Write(AO_Command_1_Register,0x2000);

 /*
 * Writing to register Interrupt_B_Enable_Register with address 75.
 *    AO_BC_TC_Interrupt_Enable <= 0
 *    AO_START1_Interrupt_Enable <= 0
 *    AO_UPDATE_Interrupt_Enable <= 0
 *    AO_START_Interrupt_Enable <= 0
 *    AO_STOP_Interrupt_Enable <= 0
 *    AO_Error_Interrupt_Enable <= 0
 *    AO_UC_TC_Interrupt_Enable <= 0
 *    AO_FIFO_Interrupt_Enable <= 0
 * New pattern = 0x0000
```

```
 */
 DAQ_STC_Windowed_Mode_Write(Interrupt_B_Enable_Register,0x0000);

 /*
 * Writing to register AO_Personal_Register with address 78.
 *   AO_BC_Source_Select <= 1
 * New pattern = 0x0010
 */
 DAQ_STC_Windowed_Mode_Write(AO_Personal_Register,0x0010);

 /*
 * Writing to register Interrupt_B_Ack_Register with address 3.
 *   AO_BC_TC_Trigger_Error_Confirm <= 1
 *   AO_BC_TC_Error_Confirm <= 1
 *   AO_UC_TC_Interrupt_Ack <= 1
 *   AO_BC_TC_Interrupt_Ack <= 1
 *   AO_START1_Interrupt_Ack <= 1
 *   AO_UPDATE_Interrupt_Ack <= 1
 *   AO_START_Interrupt_Ack <= 1
 *   AO_STOP_Interrupt_Ack <= 1
 *   AO_Error_Interrupt_Ack <= 1
 * New pattern = 0x3F98
 */
 DAQ_STC_Windowed_Mode_Write(Interrupt_B_Ack_Register,0x3F98);

  /*
 * Writing to register Joint_Reset_Register with address 72.
 *   AO_Configuration_Start <= 0
 *   AO_Configuration_End <= 1
 * New pattern = 0x0200
 */
 DAQ_STC_Windowed_Mode_Write(Joint_Reset_Register,0x0200);
}

void AO_Board_Personalize()
{
 /*
 * Writing to register Joint_Reset_Register with address 72.
 *   AO_Configuration_Start <= 1
 * New pattern = 0x0020
 */
 DAQ_STC_Windowed_Mode_Write(Joint_Reset_Register,0x0020);

 /*
 * Writing to register AO_Personal_Register with address 78.
 *   AO_Fast_CPU <= p->ao_fast_cpu (0)
 *   AO_UPDATE_Pulse_Timebase <= p->ao_update_pulse_timebase (0)
 *   AO_UPDATE_Pulse_Width <= p->ao_update_pulse_width (1)
 *   AO_DMA_PIO_Control <= p->ao_dma_pio_control (0)
 *   AO_AOFREQ_Polarity <= p->ao_dafreq_polarity (0)
 *   AO_TMRDACWR_Pulse_Width <= p->ao_tmrdacwr_pulse_width (1)
 *   AO_FIFO_Enable <= p->ao_fifo_enable (1)
 *   AO_FIFO_Flags_Polarity <= p->ao_fifo_flags_polarity (0)
```

```
 *   AO_Number_Of_DAC_Packages <= p->ao_number_of_dac_packages (0)
 * New pattern = 0x1430
 */
 DAQ_STC_Windowed_Mode_Write(AO_Personal_Register,0x1430);

 /*
 * Writing to register Clock_and_FOUT_Register with address 56.
 *   AO_Source_Divide_By_2 <= p->ao_source_divide_by_2 (0)
 *   AO_Output_Divide_By_2 <= p->ao_output_divide_by_2 (1)
 * New pattern = 0x1B20
 */
 DAQ_STC_Windowed_Mode_Write(Clock_and_FOUT_Register,0x1B20);

 /*
 * Writing to register AO_Output_Control_Register with address 86.
 *   AO_UPDATE_Output_Select <= p->ao_update_output_select (0)
 * New pattern = 0x0000
 */
 DAQ_STC_Windowed_Mode_Write(AO_Output_Control_Register,0x0000);

 /*
 * Writing to register AO_START_Select_Register with address 66.
 *   AO_AOFREQ_Enable <= p->ao_dafreq_enable (0)
 * New pattern = 0x0000
 */
 DAQ_STC_Windowed_Mode_Write(AO_START_Select_Register,0x0000);

 /*
 * Writing to register Joint_Reset_Register with address 72.
 *   AO_Configuration_Start <= 0
 *   AO_Configuration_End <= 1
 * New pattern = 0x0200
 */
 DAQ_STC_Windowed_Mode_Write(Joint_Reset_Register,0x0200);

}

/*
* Use this function to set the source and update mode for the LDAC<0..1>
* signals.
*/

void AO_LDAC_Source_And_Update_Mode()
{

 /*
 * Writing to register Joint_Reset_Register with address 72.
 *   AO_Configuration_Start <= 1
 * New pattern = 0x0020
 */
 DAQ_STC_Windowed_Mode_Write(Joint_Reset_Register,0x0020);

 /*
```

```
 * Writing to register AO_Command_1_Register with address 9.
 *   AO_LDAC0_Source_Select <= p->ao_ldac0_source_select (0)
 *   AO_DAC0_Update_Mode <= p->ao_dac0_update_mode (0)
 *   AO_LDAC1_Source_Select <= p->ao_ldac1_source_select (0)
 *   AO_DAC1_Update_Mode <= p->ao_dac1_update_mode (0)
 * New pattern = 0x0014
 */
 DAQ_STC_Windowed_Mode_Write(AO_Command_1_Register,0x0000);

 /*
 * Writing to register Joint_Reset_Register with address 72.
 *   AO_Configuration_Start <= 0
 *   AO_Configuration_End <= 1
 * New pattern = 0x0200
 */
 DAQ_STC_Windowed_Mode_Write(Joint_Reset_Register,0x0200);

}

//TITLE: DAC_OUT
void DAC_OUT(long double uVolts) {

    long double  tempV;
    unsigned int scaledU=0;

    scaledU = (unsigned int) (uVolts / 10 * 32768);
    Board_Write(AO_DAC_1_Data_Register, scaledU);
     return;

}



/* This file contains the fuctions necessary to configure the PNP *
 * for the E-Series boards.  The PNP configuration is made up of  *
 * six functions:  Write_PNP_Initiation_Key, Reset_PNP_Registers, *
 * Set_PNP_Isolation_State, Assign_PNP_Base_Address, and Activate_*
 * PNP_Board.  Two extra functions, Assign_PNP_Data_Port_Address, *
 * and Verify_PNP_Base Address, are added to confirm that the    *
 * board was properly configured.  The low level functions Write_ *
 * PNP_Address_Port_Byte, and Write_PNP_Command_Port_Byte write   *
 * data to the address and command registers, respectively       */


#include <hw/inout.h>

extern int Board_Base_Address, Read_Port_Address;
int Read_PNP_Data_Port_Byte(void);
void Write_PNP_Command_Port_Byte(int iData);
void Write_PNP_Initiation_Key();
void Reset_PNP_Registers();
void Set_PNP_Isolation_State();
```

```
/* Write data to the PNP address port.  Data is one byte.        */

void Write_PNP_Address_Port_Byte(int iData)
{

    out8(0x279, iData);
}


/* Write to the PNP command port.  Data is one byte.                */

void Write_PNP_Command_Port_Byte(int iData)
{

    out8(0xA79, iData);
}

int Read_PNP_Data_Port_Byte()
{
    int iData;

    iData = in8(Read_Port_Address);
    return(iData);
}


void Write_PNP_Initiation_Key()
{

    Write_PNP_Address_Port_Byte(0x00);
    Write_PNP_Address_Port_Byte(0x00);
    Write_PNP_Address_Port_Byte(0x6A);
    Write_PNP_Address_Port_Byte(0xB5);
    Write_PNP_Address_Port_Byte(0xDA);
    Write_PNP_Address_Port_Byte(0xED);
    Write_PNP_Address_Port_Byte(0xF6);
    Write_PNP_Address_Port_Byte(0xFB);
    Write_PNP_Address_Port_Byte(0x7D);
    Write_PNP_Address_Port_Byte(0xBE);
    Write_PNP_Address_Port_Byte(0xDF);
    Write_PNP_Address_Port_Byte(0x6F);
    Write_PNP_Address_Port_Byte(0x37);
    Write_PNP_Address_Port_Byte(0x1B);
    Write_PNP_Address_Port_Byte(0x0D);
    Write_PNP_Address_Port_Byte(0x86);
    Write_PNP_Address_Port_Byte(0xC3);
    Write_PNP_Address_Port_Byte(0x61);
    Write_PNP_Address_Port_Byte(0xB0);
    Write_PNP_Address_Port_Byte(0x58);
    Write_PNP_Address_Port_Byte(0x2C);
```

```
    Write_PNP_Address_Port_Byte(0x16);
    Write_PNP_Address_Port_Byte(0x8B);
    Write_PNP_Address_Port_Byte(0x45);
    Write_PNP_Address_Port_Byte(0xA2);
    Write_PNP_Address_Port_Byte(0xD1);
    Write_PNP_Address_Port_Byte(0xE8);
    Write_PNP_Address_Port_Byte(0x74);
    Write_PNP_Address_Port_Byte(0x3A);
    Write_PNP_Address_Port_Byte(0x9D);
    Write_PNP_Address_Port_Byte(0xCE);
    Write_PNP_Address_Port_Byte(0xE7);
    Write_PNP_Address_Port_Byte(0x73);
    Write_PNP_Address_Port_Byte(0x39);
    return;
}

void Reset_PNP_Registers()
{

  Write_PNP_Address_Port_Byte(0x02);
    Write_PNP_Command_Port_Byte(0x07);
    return;
}

void Set_PNP_Isolation_State()
{

    Write_PNP_Address_Port_Byte(0x03);
    Write_PNP_Command_Port_Byte(0x00);
    return;
}

void Set_PNP_Configuration_State()
{

    Write_PNP_Address_Port_Byte(0x06);
    Write_PNP_Command_Port_Byte(0x01);
    return;
}

void Assign_PNP_Base_Address()
{
  int iHi_Base_Address, iLo_Base_Address;

  iLo_Base_Address = Board_Base_Address & 0xff;
  iHi_Base_Address = (Board_Base_Address >> 8) & 0xff;


    Write_PNP_Address_Port_Byte(0x60);          /* Set the upper byte of the base address  */
    Write_PNP_Command_Port_Byte(iHi_Base_Address);
    Write_PNP_Address_Port_Byte(0x61);          /* Set the lower byte of the base address  */
    Write_PNP_Command_Port_Byte(iLo_Base_Address);
    return;
```

```
}

void Assign_PNP_Data_Port_Address()
{

    Write_PNP_Address_Port_Byte(0x00);
    Write_PNP_Command_Port_Byte(Read_Port_Address>>2);
    return;
}

int Verify_PNP_Base_Address()
{

    int iHi_Base_Address, iLo_Base_Address, iHi_Read, iLo_Read;

  iLo_Base_Address = Board_Base_Address & 0xff;
  iHi_Base_Address = (Board_Base_Address >> 8) & 0xff;

    Write_PNP_Address_Port_Byte(0x60);          /* Read the upper byte of the base address  */
    iHi_Read = Read_PNP_Data_Port_Byte();
    Write_PNP_Address_Port_Byte(0x61);          /* Read the lower byte of the base address  */
    iLo_Read = Read_PNP_Data_Port_Byte();

    if (iLo_Read == iLo_Base_Address && iHi_Read == iHi_Base_Address)
      return(0);
    return(1);
}

void Activate_PNP_Board()
{

    Write_PNP_Address_Port_Byte(0x03);
    Write_PNP_Command_Port_Byte(1);
    Write_PNP_Address_Port_Byte(0x07);
    Write_PNP_Command_Port_Byte(0x00);
    Write_PNP_Address_Port_Byte(0x30);
    Write_PNP_Command_Port_Byte(0x01);
    Write_PNP_Address_Port_Byte(0x07);
    Write_PNP_Command_Port_Byte(0x01);
    Write_PNP_Address_Port_Byte(0x30);
    Write_PNP_Command_Port_Byte(0x01);
    Write_PNP_Address_Port_Byte(0x02);
    Write_PNP_Command_Port_Byte(0x02);
  return;
}

int Setup_PNP_Board()
{
    int iError;

    Write_PNP_Initiation_Key();
    Reset_PNP_Registers();
    Write_PNP_Initiation_Key();
```

```
    Set_PNP_Isolation_State();
    Set_PNP_Configuration_State();
    Assign_PNP_Base_Address();
    Assign_PNP_Data_Port_Address();
    iError = Verify_PNP_Base_Address();
    if (iError)
            return(1);
    Activate_PNP_Board();
    return(0);
}
```

```
/* file global.h
 * global declarations
******************/
#include<sys/neutrino.h>
#include<hw/inout.h>
#include <sys/siginfo.h>
#include <signal.h>
#include <time.h>
#include<stdio.h>
#include<process.h>
#include<math.h>
#include<stdlib.h>
#include <pthread.h>
#include <sched.h>
#include <errno.h>


/*Local constants/function defs*/
#define DAC_gain        2.2

#define dataLength      500
#define dataWidth       6
#define SGN(x)          ((x) == 0.0 ? (0.0) : ((x) > (0.0) ? (1.0) :(-1.0)))
#define ROUND(x)        ((((int)(10.0*x))-(10*((int)(x))))>5 ? ((int)(x)+1):((int)(x)))
#define PI              3.14159265359
#define INTR            8
#define cartLimit       0.25  /*cart movement limit .25m*/
#define angleLimit      30    /*pendulum movement limit degrees*/
#define currentLimit    (9.99*DAC_gain) /*limit om motor current from the amp (amps) */
#define percentOver     3.0   /*percent over motor current permitted by contol signal*/


extern long double sampleRateHz;
extern long double samplePeriod;
```

```
/*define structures and typeefs */
struct threebyte {
      unsigned char  lsb;
      unsigned char  lsb_1;
      unsigned char  msb;
     };

struct twobyte {
      unsigned char  lsb;
      unsigned char  msb;
     };

struct encReference {
       long int aRef;
       long int bRef;
       long int cRef;
       long int dRef;
     };

typedef union tByte {
       unsigned long int n;
       struct threebyte tb;
      }tByte;

typedef union twoByte {
       int n;
       struct twobyte tb;
      }twoByte;

/*Global variables */
extern struct encReference initValues;
extern long double outputArray[dataLength][dataWidth];
extern long double sampleRateHz;
extern unsigned count;
extern float floatVal[2000];
extern int l;
extern FILE *f_in, *fshake, *fstand;
extern pthread_mutex_t mutex; //mutex used for sync. access to feedback gain coefs.
extern long double G_new[4]; //for gain coefs sent by client


/*Function prototypes */
void DAC_OUT(long double uVolts);
void initHardware(void);
void initEncoder(int ip, int mcr, int icr, int occr, int qr, short bAddr);
void writeIp(int, int, short);
int readIp(int, short);
void loadCntr(int, int, short);
int readCntr(int, short);
void loadPr(int, int, short);
int readOl(int, short);
void* controlThread(void*);
```

```
void* serverThread(void*);
void finish();
```

Enclosed CD-ROM

The enclosed CD-ROM contains a 30 second movie presenting the pendulum swing-up and stabilization experiment held with the pendulum apparatus. The movie is shot in MPEG format with Sony® digital camera and can be viewed with RealNetworks® Real Player™, Microsoft® Media Player™ or any MPEG player (not included) .