MECHANICAL DESIGN
AND
CONTROL OF THE PENDUBOT

BY

DANIEL JEROME BLOCK

B.S., University of Illinois, 1991

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in General Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1996

Urbana, Illinois

MECHANICAL DESIGN
AND
CONTROL OF THE PENDUBOT

Daniel Jerome Block, M.S.
Department of General Engineering
University of Illinois at Urbana-Champaign, 1996
Mark W. Spong, Adviser

In this thesis we present the design and control of an underactuated two link robot called the Pendubot. We first give details of the design of the Pendubot, discussing the components of the linkage and the interface to the PC used as the controller. Parameter identification of the Pendubot is accomplished by solid modeling methods, a least squares solution to the energy equations of the linkage, and a constrained minimization technique. With the identified parameters, mathematical models are developed to facilitate controller design. The goal of the control is to swing the linkage from the downward equilibrium position to the unstable inverted equilibrium positions and balance it there. Two control algorithms are used for this task. Partial feedback linearization techniques are used to design the swing up control. The balancing control is then designed by linearizing the dynamic equations about the desired equilibrium point. LQR and pole placement techniques are used to design the stabilizing controller.

DEDICATION

I dedicate this work to my wife, Faith, who has been my number one fan and who will always be my number one hero.

ACKNOWLEDGMENTS

There are many individuals that I would like to thank that played a role in this thesis work. First and for most is my adviser, Professor Mark W. Spong, who came up with the concept of the Pendubot. His knowledge and guidance is what made this thesis work a success.

Special thanks go to Dave Roberts who helped me in the mechanical design of the Pendubot. His suggestions and ideas and especially his skillful hands in building the Pendubot are greatly appreciated.

Thanks to my father, Rev. George Gude, who took the photographs seen in the thesis. Phil Walsh who helped me in filtering the Pendubot's velocity data. Also to Dave Ruder who came up with the name "Pendubot" and helped me video tape my work.

Finally, I would like to thank my entire family for their love and support throughout my graduate school days. They and especially my wife, Faith, gave me the encouragement to complete this work.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

1. INTRODUCTION

At the University of Illinois, extensive research and development has gone into the concept and design of the two link underactuated planar robot called the Acrobot [1]. To extend this research in underactuated planar linkages, we came up the concept of the Pendubot [10], [11]. It is a counterpart of the Acrobot in that its two links are mounted vertically, but instead of having the actuation at its elbow, the Pendubot is driven at its shoulder joint. This makes for a slightly simpler control design when compared with the Acrobot, but all similar control issues can be studied and implemented. The goal of the Pendubot controller is to swing the mechanism from its open loop stable configuration to the unstable equilibrium points and then to catch the unactuated link and balance it there.

In chapter two we explain the components that were designed or purchased to assemble the Pendubot. Chapter three quickly goes through the derivation of the mathematical model of the Pendubot. The ordinary differential equations (O.D.E.s) found in this chapter are the basis for the controller designs to be used.

Chapter four explains the parameter identification methods used to identify the Pendubot's actual dynamic parameters. The first method is off-line and uses a CAD software package to draw a solid model of the links. Then by specifying the density of the material used for each component, the CAD package is able to calculate the parameters of the links. The second method is on-line and uses the energy equations of the linkage to form a least squares problem that can be solved for the unknown parameters. The third method, which is also on-line, solves a constrained minimization problem that finds a best fit for the parameters by minimizing the error between actual response data and simulated response data. The advantage of both these methods compared to differential methods [7] is that they do not require the realization of acceleration.

The next three chapters (5,6 and 7) discuss the control algorithms used to swing up and balance the links at unstable equilibrium points. For the swing up control we use the method of partial feedback linearization discussed in [2] and [3]. The balancing control was then found by linearizing the system and designing a full state feedback controller for that linearized model.

Chapters 8 and 9 display the results of the implemented controllers. Chapter 8 has the simulated results which can be compared with the actual responses of the Pendubot system shown in Chapter 9.

Appendix A demonstrates how friction can be added to both the mathematical model of the Pendubot and the energy equation identification scheme. Appendix B reproduces the derived equations that linearize the Pendubot about any desired equilibrium point. Appendix C lists the Matlab m-files needed to perform the minimization method. Appendix D lists the simulation files used to simulate the Pendubot with the software package Simnon [4]. Appendix E lists the source code of three implemented controllers. Finally, Appendix F reproduces the mechanical drawings of the components of the linkage.

2. MECHANICAL DESIGN AND CONTROLLER INTERFACE

The Pendubot is shown schematically in Figure 2.1 and a photograph of the Pendubot in its upright, "top", balancing position is shown in Figure 2.2. The actuated joint is driven by a high torque 90VDC permanent magnet motor without gear reduction. To give joint one

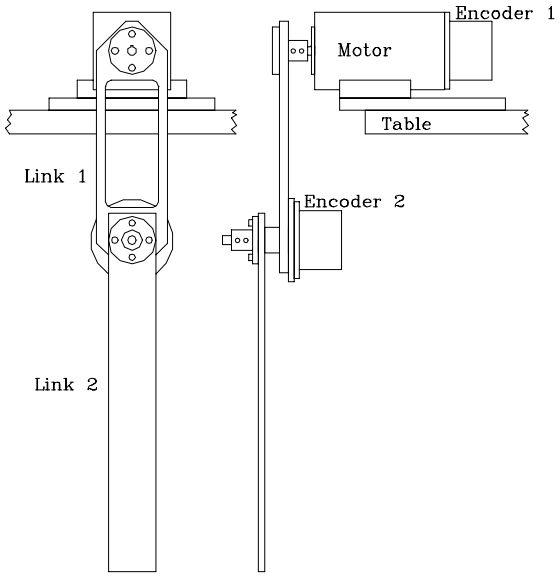

Figure 2.1 Front and Side Perspective Drawings of the Pendubot.

direct drive control, we designed the Pendubot to hang off the side of a table coupling link one directly to the shaft of the motor. The mount and bearings of the motor are then the support for the entire system. Link one also includes the bearing housing which allows for the motion of joint two. Needle roller bearings riding on a ground shaft were used to construct this revolute joint for joint two. The shaft extends out both directions of the housing allowing coupling to both link two and an optical encoder mounted on link one. This optical encoder produces the position feedback of link two. The design gives both links full 360° of motion. Link one, however, cannot continuously rotate due to the encoder cable for link two. Link two has no constraint on continuous revolutions.

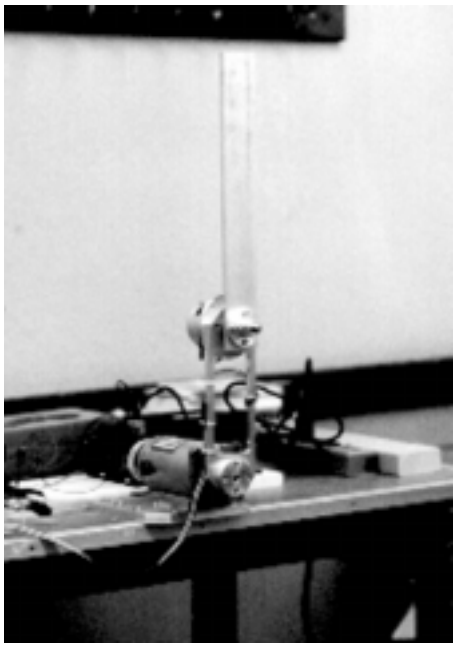

Figure 2.2 Photograph of the Pendubot in its Top Balancing Position.

Link two is simply a ¼ inch (0.635 cm) thick length of aluminum with a coupling that attaches to the shaft of joint two.

The lengths of the links were determined by intuition and earlier work on the Acrobot [1] and then confirmed with simulation. The intuition comes from thinking about balancing a broom or a similar object in the palm of your hand. The longer the broom the easier it is too balance. Of course if it gets too long it is too heavy to hold and in the case of the Pendubot even harder to swing up from the hanging position. A length of 14 inches (35.56 cm) was chosen for link two. This gave it a good center of mass location with acceptable total mass.

Designing the length of link one is a little different. It needs to have some length and good stiffness so it can quickly get under link two when balancing, but the heavier it is the more torque the motor must produce. A length of 8 inches (20.32cm) was chosen for link one and the center material of the link was removed (See Figure 2.1). Please refer to Appendix F for the mechanical drawings of the links and couplings.

To test our intuition on the length and weights of the links, we first performed simulation studies of the Pendubot and its controller. AutoCAD's solid modeling extension was used to get an approximation of the dynamic parameters of the system (See Chapter 4). Then the software package Simnon [4] was used to simulate the dynamic equations and

controller of the Pendubot (See Chapter 8). The design was confirmed by observing that the control effort remained less than the maximum torque of the motor when swinging the links to their upright position and balancing them there.

The final component of the Pendubot's hardware is its controller. See Figure 2.3 for a pictorial description of the interface between the Pendubot and the controller. BEI 1024 counts/rev resolution optical encoders, one attached at the elbow joint and one attached to the motor, were used as the feedback mechanism for the joint angles. Advanced Motion Control's 25A8 PWM servo amplifier was used to drive the motor. In the control algorithm this amplifier can be thought of as just a gain. In the case of the Pendubot we setup the amplifier in torque mode and adjusted it for a gain of 1V=1.2Amps.

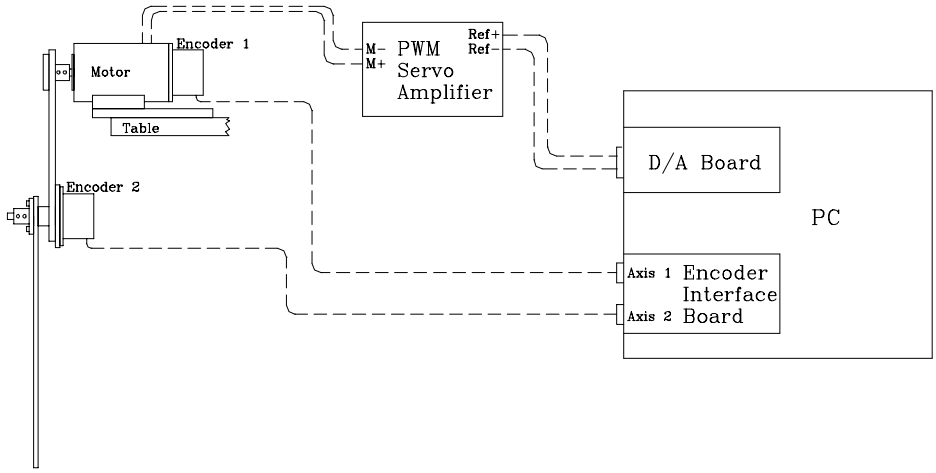

Figure 2.3 Pictorial of the Pendubot's Interface with its Controller.

In an attempt to simplify the controller for the Pendubot and minimize its cost, we implemented our control algorithm using only the microprocessor in our PC instead of purchasing an additional DSP card. We used a 486DX2/50 IBM compatible PC with a D/A card and an encoder interface card. The DAC-02 card by Keithley Metrabyte was used for the digital to analog converter and the 5312B by Technology 80 was used to interface with the optical encoders. The X4 quadrature mode was used on this card to increase the resolution of the optical encoders by 4, giving 4096 counts/rev. Then with the software library routines accompanying the cards, we were able to write C programs to implement the control algorithms (See Appendix E).

The only difficulty in using a PC as a digital controller is finding a way to reliably get a fast sampling period interrupt. DOS does produce a clock pulse but it only occurs every 55 milliseconds making it useless for this system which needs at least a 10 ms sampling period. To get around this, the timer chip on the motherboard of the PC was directly programmed to achieve a higher resolution. The software package "PC Timer Tools" by Ryle Design includes an alarm algorithm that can be used to produce an appropriate sampling period (i.e. 5ms). The format of the control algorithm then is as follows:

```
/* Perform all needed initializations */
/* start 5ms alarm */
while (Continue_Control==TRUE) {
   /* sample encoder positions */
   /*  use finite differences to calculate velocity */
   /* calculate needed control effort */
   /* output control value to motor */
   while (Alarm_expired == FALSE) {
     /* continue to loop until alarm expires */
   } /***** end of second while *****/
}  /***** end of first while ****/
```

This control design worked very well. We are able to reliably achieve a 1ms sampling period even when computing the inverse dynamic equations for the partial feedback linearization control (See Chapter 5). A 5 ms sampling period was used by most of our controllers in order to allow us to save response data while the controller was operating. A 5 ms sampling period also allows room to update our controller with a Windows GUI interface which requires more overhead running in the Windows operating system.

3. SYSTEM MODEL

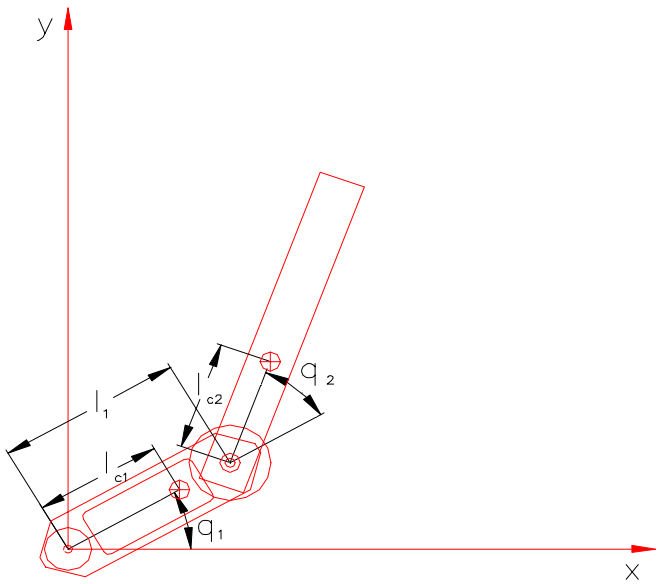

Figure 3.1 Coordinate Description of the Pendubot. l_1 is the length of link one, l_{c1} and l_{c2} are the distances to the center of mass of the respective links and q_1 and q_2 are the joint angles of the respective links.

The equations of motion for the Pendubot can be found using Lagrangian dynamics [5]. In matrix form the equations are

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{3.1}$$

where τ is the vector of torque applied to the links and **q** is the vector of joint angle positions with

$$D(q) = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \qquad \begin{aligned} d_{11} &= m_1 l_{c1}^2 + m_2 (l_1^2 + l_{c2}^2 + 2 l_1 l_{c2} \cos q_2) + I_1 + I_2 \\ d_{12} &= d_{21} = m_2 (l_{c2}^2 + l_1 l_{c2} \cos q_2) + I_2 \\ d_{22} &= m_2 l_{c2}^2 + I_2 \end{aligned} \tag{3.2}$$

and

$$C(q,\dot{q}) = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ -h\dot{q}_1 & 0 \end{bmatrix} \qquad h = -m_2 l_1 l_{c2} \sin q_2 \tag{3.3}$$

and

$$g(q) = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}$$

$$\begin{aligned} \phi_1 &= (m_1 l_{c1} + m_2 l_1) g \cos q_1 + m_2 l_{c2} g \cos(q_1 + q_2) \\ \phi_2 &= m_2 g l_{c2} \cos(q_1 + q_2) \end{aligned} . \tag{3.4}$$

m_1	:	the total mass of link one.
l_1	:	the length of link one (See Figure 3.1).
l_{c1}	:	the distance to the center of mass of link 1 (See Figure 3.1).
I_1	:	the moment of inertia of link one about its centroid.
m_2	:	the total mass of link two.
l_{c2}	:	the distance to the center of mass of link 2 (See Figure 3.1).
I_2	:	the moment of inertia of link two about its centroid.
g	:	the acceleration of gravity.

From the above equations it is observed that the seven dynamic parameters can be grouped into the following five parameter equations

$$\begin{aligned} \theta_1 &= m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \\ \theta_2 &= m_2 l_{c2}^2 + I_2 \\ \theta_3 &= m_2 l_1 l_{c2} \\ \theta_4 &= m_1 l_{c1} + m_2 l_1 \\ \theta_5 &= m_2 l_{c2} \end{aligned} . \tag{3.5}$$

For a control design that neglects friction, these five parameters are all that are needed. There is no reason to go a step further and find the individual parameters since the control equations can be written with only the five parameters. Substituting these parameters into the above equations leaves the following matrices

$$D(q) = \begin{bmatrix} \theta_1 + \theta_2 + 2\theta_3 \cos q_2 & \theta_2 + \theta_3 \cos q_2 \\ \theta_2 + \theta_3 \cos q_2 & \theta_2 \end{bmatrix}, \tag{3.6}$$

$$C(q, \dot{q}) = \begin{bmatrix} -\theta_3 \sin(q_2)\dot{q}_2 & -\theta_3 \sin(q_2)\dot{q}_2 - \theta_3 \sin(q_2)\dot{q}_1 \\ \theta_3 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix}, \tag{3.7}$$

$$g(q) = \begin{bmatrix} \theta_4 g \cos q_1 + \theta_5 g \cos(q_1 + q_2) \\ \theta_5 g \cos(q_1 + q_2) \end{bmatrix}. \tag{3.8}$$

Finally, using the invertible property of the mass matrix D(q) [5], the state equations are given by

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = D(q)^{-1}\tau - D(q)^{-1}C(q,\dot{q})\dot{q} - D(q)^{-1}g(q)$$

$$x_1 = q_1, x_2 = \dot{q}_1, x_3 = q_2, x_4 = \dot{q}_2$$

$$\dot{x}_1 = x_2 \qquad . \qquad (3.9)$$

$$\dot{x}_2 = \ddot{q}_1$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = \ddot{q}_2$$

4. SYSTEM IDENTIFICATION

4.1 CAD Solid Model

After formulating the mathematical model of the Pendubot, the next step was to identify the five parameters in equation (3.5). An AutoCAD solid model of the Pendubot was drawn to give approximate numbers for these parameters. As mentioned previously, these approximate parameters helped in the design of the Pendubot. Used in simulations, they allowed us to determine if the motor would be powerful enough to manipulate the two links. They also served as a guide to determine the accuracy of the on-line identification methods described in the next two sections. Taking into account the amplifier gain, Kamp =1.2A/V, and the torque constant of the motor, K_T =3.546 lbin/A (0.4006 Nm/A), the solid model parameters were

$\theta_1 = 0.089252$ V*s^2
$\theta_2 = 0.027630$ V*s^2
$\theta_3 = 0.023502$ V*s^2
$\theta_4 = 0.011204$ V*s^2/in (0.44110 V*s^2/m)
$\theta_5 = 0.002938$ V*s^2/in (0.11567 V*s^2/m).

4.2 Energy Equation Method

This on-line identification scheme uses the energy theorem to form equations that can be solved for the unknown parameters by a least squares problem [6].

The kinetic energy of the Pendubot is written as

$$K = \frac{1}{2}\dot{q}D(q)\dot{q} \tag{4.1}$$

where D(q) is defined by equation (3.6). Performing the matrix multiplication produces the following equation for the kinetic energy

$$K = \tfrac{1}{2}\dot{q}_1^2\theta_1 + (\tfrac{1}{2}\dot{q}_1^2 + \dot{q}_1\dot{q}_2 + \tfrac{1}{2}\dot{q}_2^2)\theta_2 + (\cos q_2\dot{q}_1^2 + \cos q_2\dot{q}_1\dot{q}_2)\theta_3. \tag{4.2}$$

The potential energy of the Pendubot is written

$$V = (m_1 l_{c1} + m_2 l_1) g \sin q_1 + m_2 l_{c2} g \sin(q_1 + q_2). \tag{4.3}$$

In terms of the parameters to be identify it is simplified to

$$V = \theta_4 g \sin q_1 + \theta_5 g \sin(q_1 + q_2). \tag{4.4}$$

Looking at the above equations it is observed that the kinetic and potential energy equations are both linear in the inertial parameters. A simple way to write these equations then is

$$K = \sum_{i=1}^{5} \frac{\partial K}{\partial \theta_i}\theta_i = \sum_{i=1}^{5} DK_i \theta_i$$
$$V = \sum_{i=1}^{5} \frac{\partial V}{\partial \theta_i}\theta_i = \sum_{i=1}^{5} DV_i \theta_i . \tag{4.5}$$

For the Pendubot the new terms DK and DV are

$$\begin{aligned} DK_1 &= \tfrac{1}{2}\dot{q}_1^2 \\ DK_2 &= \tfrac{1}{2}\dot{q}_1^2 + \dot{q}_1\dot{q}_2 + \tfrac{1}{2}\dot{q}_2^2 \\ DK_3 &= \cos q_2 \dot{q}_1^2 + \cos q_2 \dot{q}_1 \dot{q}_2 , \\ DK_4 &= 0 \\ DK_5 &= 0 \end{aligned} \tag{4.6}$$

$$\begin{aligned} DV_1 &= 0 \\ DV_2 &= 0 \\ DV_3 &= 0 \\ DV_4 &= g \sin q_1 \\ DV_5 &= g \sin(q_1 + q_2) \end{aligned} \quad . \tag{4.7}$$

The energy theorem which states that the work of forces applied to a system is equal to the change of the total energy of the system can be written as

$$\int_{t_1}^{t_2} \mathbf{T}^T \dot{\mathbf{q}} dt = (K(t_2) + V(t_2)) - (K(t_1) + V(t_1)) = L(t_2) - L(t_1) \tag{4.8}$$

where $L(t_i)$ is the total energy at time t_i, $L(t_i) = K(t_i) + V(t_i)$, and **T** is the vector of torque applied at the joints. **T** includes both the motor torque and the friction forces and can be written

$$\mathbf{T} = \tau + \Gamma_f. \tag{4.9}$$

For this study we neglected friction setting Γ_f to zero. See Appendix A for the addition of friction terms.

Again using the property that K and V are linear in the inertial parameters, the difference in the total energy is defined $L(t_2) - L(t_1) = \mathbf{DL}^T\boldsymbol{\theta}$, where

$$\mathbf{DL}^T = [DL_1(t_2)-DL_1(t_1) \quad \ldots \quad DL_5(t_2)-DL_5(t_1)] \tag{4.10}$$

and

$$DL_i(t_k) = DK_i(t_k)+DV_i(t_k). \tag{4.11}$$

This leaves the energy equation in the form

$$\int_{t_1}^{t_2} \mathbf{T}^T \dot{\mathbf{q}} dt = \mathbf{D}\mathbf{L}^T \theta . \tag{4.12}$$

Defining a new vector d, $d^T = \mathbf{D}\mathbf{L}^T$, the K^{th} equation related to the time interval (t_{K-1},t_K) is written

$$(\int_{t_{K-1}}^{t_K} \mathbf{T}^T \dot{\mathbf{q}} dt)_K = d_K^T \theta . \tag{4.13}$$

The K equations can be combined into a standard over determined matrix equation, Ax=b, and solved by least squares techniques. Also since $DL_i(0) = 0$ for (i=1,...,5), we can write this equation as

$$(\int_{t_0}^{t_K} \mathbf{T}^T \dot{\mathbf{q}} dt)_K = d_K^T \theta \tag{4.14}$$

where the K^{th} equation is now for the time interval (t_0,t_K).

To implement this identification scheme we wrote a simple program that drove the Pendubot with an open loop signal and at the same time recorded the response of the system. This response data was then loaded into Matlab where the identification algorithm could be performed. To approximate the integral on the left hand side of the least squares problem the backwards trapezoidal rule was used. The resulting Matlab M-file was as follows:

```
%q1, dq1, q2 and dq2 are vectors of joint positions and velocities
g=386; (SI Units = 9.8)
dL1 = (.5*dq1.^2);
dL2 = (.5*dq1.^2 + dq1.*dq2 + .5*dq2.^2);
dL3 = (cos(q2).*(dq1.^2 + dq1.*dq2));
dL4 = (g*sin(q1));
dL5 = (g*sin(q1+q2));
taudq1 = tau.*dq1;       %tau is the open loop control effort
for i = 1:(length(dL1)-10),
DL(i,1) = dL1(i+10)-dL1(1);
DL(i,2) = dL2(i+10)-dL2(1);
DL(i,3) = dL3(i+10)-dL3(1);
DL(i,4) = dL4(i+10)-dL4(1);
DL(i,5) = dL5(i+10)-dL5(1);
Itq(i,1) =  trapz(t(1:i+10,1),taudq1(1:i+10,1));
end
theta = nnls(DL,Itq)   %non-negative least squares solution to Ax=b.
```

Different open-loop inputs (i.e. sine wave, square wave, single steps) were tried in an attempt to see which best identified the system. A simple step input was found to work well giving the most consistent results. The input units were volts (V) applied to the amplifier in

order that the parameters identified also contained the amplifier gain. The mean of the parameters found by this method with the step input, v = 2.5, were

$\theta_1 = 0.0799$ V*s^2
$\theta_2 = 0.0244$ V*s^2
$\theta_3 = 0.0205$ V*s^2
$\theta_4 = 0.0107$ V*s^2/in (0.42126 V*s^2/m)
$\theta_5 = 0.0027$ V*s^2/in (0.10630 V*s^2/m).

We did attempt to add the friction components to the identification algorithm. Unfortunately, we were unable to find conclusive results for the friction terms. Please see Appendix A for the addition of the friction to the Pendubot model and this identification method. The friction in the Pendubot system is low which may be the reason the energy based identification algorithm does not identify it well. As pointed out in Prüfer, Schmidt and Wahl [7] the energy based algorithm generally has difficulty identifying friction terms. Fortunately the parameters found ignoring friction, as we will demonstrate, work very well in controlling the system.

4.3 Optimization Method

The second on-line method implemented to identify the unknown parameters of the Pendubot uses a constrained minimization algorithm. The error between actual data collected from the Pendubot and data created by simulations is minimized by varying the unknown parameters until a best fit is found. The minimization problem can be written as follows

$$Min_{\theta}\left\{\sum_{i=1}^{4}\int_0^t (y_{ri}-y_{si})^2\right\}$$
s.t. (4.15)
$\theta_{lb} < \theta < \theta_{ub}$

where y_{ri} is the output position and velocity data of the actual Pendubot and y_{si} is the output position and velocity data found by simulation runs. θ is the vector of inertial parameters to be found and θ_{lb} and θ_{ub} are the lower and upper bounds on these parameters.

The Matlab function "constr" [8] is used to perform this minimization algorithm. "Constr" uses a sequential quadratic programming [9] method to solve the constrained minimization problem.

To implement this identification scheme in Matlab, two function m-files were needed. An objective function to be called and minimized by "constr" and a simulation function modeling the dynamics of the Pendubot. Please refer to Appendix C for the listing of these m-files and the other initialization files used. The simulation function, "pend.m" in Appendix C, defines the nonlinear O.D.E.s of the Pendubot just as seen in equation (3.9) or equation (A.3) if friction is added. To drive the simulated linkage, the same open loop torque equation applied to the Pendubot when collecting the real time data is used to calculate the torque applied to the simulated system. With this function Matlab can simulate the dynamics of the Pendubot with its O.D.E. solver "ode45".

The objective function, "pend_obj.m" in Appendix C, takes as input the unknown parameters and outputs the integral of the error squared, equation (4.15). To accomplish this the objective function first calls “ode45” with the simulation function and the given inertial parameters. This in turn returns simulated response data. The simulated response data and the actual response data are then compared at each time interval of the actual response data, 0.005 seconds. "Ode45" does not perform equally spaced time steps when evaluating the O.D.E. so the simulated response data is interpolated to match up with the time intervals of the actual response data. The Matlab function “interp1” with the spline option is used for this purpose. Finally, using the “trapz” function to evaluate the integrals, the objective function is calculated and returned to "constr".

As with the energy equation method, section 4.2, depending on the open-loop torque trajectory applied, the minimization method was able to do a good job of identify the parameters of the Pendubot. In this case sinusoidal inputs were found to give the best results. For example an open loop input, v = 0.5sin7.7t (volts), excited the system enough to allow for an adequate identification. Note, as in the energy equation method, the units of the input signal are taken to be volts (V) applied to the amplifier so the identified parameters will contain the amplifier gain. The parameters found with the above input were

$\theta_1 = 0.09242$ V*s^2
$\theta_2 = 0.0247$ V*s^2
$\theta_3 = 0.0214$ V*s^2
$\theta_4 = 0.01184$ V*s^2/in (0.46614 V*s^2/m)
$\theta_5 = 0.00254$ V*s^2/in (0.1000 V*s^2/m).

As with the energy equation method, this method was unable to identify the friction parameters of the Pendubot. Matlab had a problem solving the minimization problem when the friction terms were added. Stack faults occurred in the middle of each attempted minimization run which indicated some type of numerical problem. Possibly since the friction is low in the linkages, the "constr" function had problems finding adequate friction parameters. It is noted though that the friction is obviously present in the actual Pendubot and will add to the five parameters found.

4.4 Comparison of the Results

Table 4.1 shows a list of each set of parameters identified by the different methods. Similar results were produced by each method, which indicated success in identifying the parameters of the Pendubot. There are some variations in the parameters between the

	Solid Model	**Energy Equation**	**Minimization**
θ_1 **(V*s^2)**	0.08925	0.0799	0.09242
θ_2 **(V*s^2)**	0.02763	0.0244	0.0247
θ_3 **(V*s^2)**	0.0235	0.0205	0.0214
θ_4 **(V*s^2/in)**	0.0112	0.0107	0.01184
θ_5 **(V*s^2/in)**	0.00294	0.0027	0.00254

Table 4.1 Comprehensive List of Identified Parameters by Method.

methods, but nothing to conclude that one of the methods were in error. Since friction is ignored in both the energy equation method and the minimization method, its effect on the system are added to the parameters identified. Ignoring the friction possibly has different effects on the identification schemes, therefore creating the differences seen in the parameters. Friction does not enter into the solid model method, therefore also creating possible causes for the differences seen. Of course there are also numerical and modeling errors that are different between the methods. The solid model method is relying on the assumptions of the drafter for accuracy in the model. Both the energy equation method and the minimization method use the data collected from the Pendubot which is inherently noisy, especially the velocity data, which is derived by a finite difference equation (See Chapter 7).

Since only these small variations were found, we were able to conclude that the actual parameters of the Pendubot were in close proximity to the parameters listed in Table 4.1.

Due to the small variation of the results found, the parameters identified by the energy equation method were chosen as the parameters to be used for the control experiments. Initially attempts were made to determine if one set of parameters out performed the others. Only small performance variations, if any, were seen when comparing the different parameter sets. For this reason only one set was chosen for consistency.

5. SWING UP CONTROL

As stated earlier the goal of the Pendubot controller is to swing the links from their stable hanging position to unstable equilibrium positions and then balance the links about that equilibrium. This control is divided into two parts; swing up control, and balancing control. The swing up control uses the method of partial feedback linearization. Many different

control algorithms could have been used to perform the swing up. In fact initially we used a PID controller servoing around only the position of link one to swing up the Pendubot. This worked fine but amplified numerical noise. Partial Feedback Linearization needs position feedback from both link one and link two but takes into account the nonlinear effects of the linkage. This creates a much cleaner control compared to a PID control that must reject the effects of the first and second link.

We will now derive the partial feedback control for the Pendubot. To see a general derivation of partial feedback linearization please refer to [2] and [3].

The equations of motion of the Pendubot are given by equation (3.1). Performing the matrix and vector multiplications, the equations of motion are written

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{11}\dot{q}_1 + c_{12}\dot{q}_2 + \phi_1 = \tau_1 \tag{5.1}$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + \phi_2 = 0\,. \tag{5.2}$$

Due to the underactuation of link two we can not linearize the dynamics of both degrees of freedom. We can, however, linearize one of the degrees of freedom. This will allow us to design an outer loop control that will track a given trajectory for the linearized degree of freedom. In the case of the Pendubot we chose to linearize about the collocated degree of freedom q_1. Equation (5.2) was solved for the angular acceleration of link two

$$\ddot{q}_2 = \frac{-d_{21}\ddot{q}_1 - c_{21}\dot{q}_1 - \phi_2}{d_{22}}. \tag{5.3}$$

This was then substituted into equation (5.1) and written as

$$\bar{d}_{11}\ddot{q}_1 + \bar{c}_{11}\dot{q}_1 + \bar{c}_{12}\dot{q}_2 + \bar{\phi}_1 = \tau_1 \tag{5.4}$$

with

$$\begin{aligned}
\bar{d}_{11} &= d_{11} - \frac{d_{12}d_{21}}{d_{22}} \\
\bar{c}_{11} &= c_{11} - \frac{d_{12}c_{21}}{d_{22}} \\
\bar{c}_{12} &= c_{12} \\
\bar{\phi}_1 &= \phi_1 - \frac{d_{12}\phi_2}{d_{22}}.
\end{aligned} \tag{5.5}$$

Then just as with the full linearization method (also called the computed torque method [5]) the inner loop control that linearizes q_1 can be defined as

$$\tau_1 = \bar{d}_{11}v_1 + \bar{c}_{11}\dot{q}_1 + \bar{c}_{12}\dot{q}_2 + \bar{\phi}_1. \tag{5.6}$$

This results in the system

$$\ddot{q}_1 = v_1 \tag{5.7}$$

$$d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + \phi_2 = -d_{21}v_1. \tag{5.8}$$

Since equation (5.7) is now linear, an outer loop control can be designed to track a given trajectory for link one. The response of link two then is given by the resulting nonlinear equation (5.8). Equation (5.8) represents internal dynamics with respect to an output $y = q_1$. The goal of the outer loop control then is to track a given trajectory for link one and at the same time excite the internal dynamics to swing link two to a balancing position. For the Pendubot we chose to use a PD with feedforward acceleration

$$v_1 = \ddot{q}_1^d + K_d(\dot{q}_1^d - \dot{q}_1) + K_p(q_1^d - q_1). \tag{5.9}$$

See Figure 5.1 for a block diagram of the swing up control.

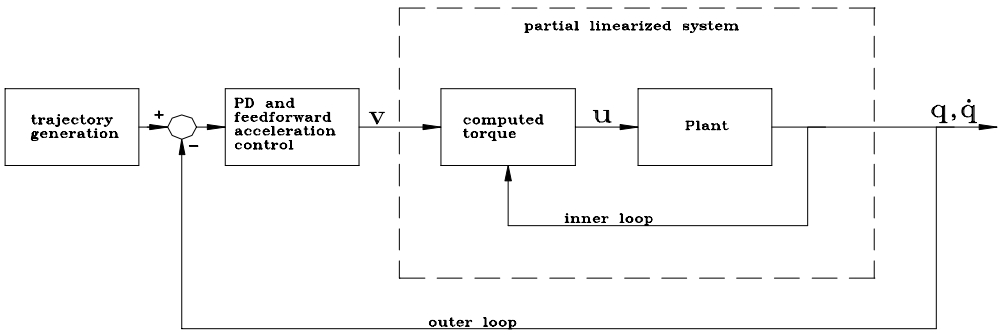

Figure 5.1 Block Diagram of the Partial Feedback Linearization Control

Now, given this controller, a trajectory must be determined to swing the links to their unstable equilibrium position. To swing the links to the upright top position ($q_1 = \pi/2$, $q_2 = 0$), we simply used a step trajectory $q_1 = \pi/2$. This trajectory worked very well in simulation but when trying it out on the actual system the starting torque of the motor was not strong enough to consistently swing the second link all the way to its upright position. On power up of the system, the first few trials would excite the second link enough to bring it to the top equilibrium position. As the motor warmed up, though, its torque constant would decrease slightly and not allow the second link to swing to the top. We also determined that the power supply for the amplifier was not a perfect match for the motor we had purchased. Only approximately 60% of the total torque of the motor was being utilized. To get around this problem we added to the swing up trajectory a small open loop step that sent the link in the negative direction for a short period of time adding potential energy into the system. This added energy allowed the motor to excite the internal dynamics and consistently swing both links to the upright position. It is noted that this torque deficit causes extended saturation in the control output. Figure 9.1 shows a plot of the voltage applied to the amplifier when performing the swing up to the top. For approximately the first half second of the swing up, the signal is saturated and therefore not performing the partial feedback linearization control. In fact, initially, the swing up control is a bang-bang control.

The swing up trajectory to swing the links to the mid position ($q_1 = -\pi/2$, $q_2 = \pi$) was a little more difficult (See Figure 5.2). Simulations were run to find a trajectory that would work well. The trajectory found for the swing up can be written

$$q_1 = 1.4\sin(5t) - \pi/2 \quad : t < 2\pi/5$$
$$q_1 = -\pi/2 \quad : t > 2\pi/5.$$

This trajectory pumps energy into the system by causing the second link to swing back and forth and finally up to its middle equilibrium point.

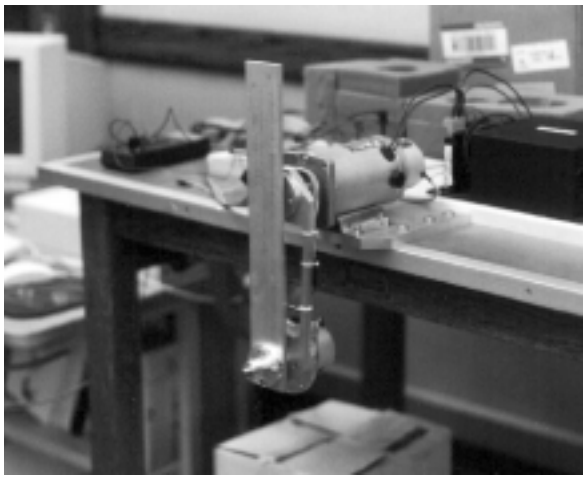

Figure 5.2 Photograph of the Pendubot in its Mid Balancing Position.

To fine tune the swing up control, the K_p and K_d gains were adjusted. Correct gains were found that swing the second link slowly into its equilibrium position so that the balancing controller can catch and balance the link.

6. BALANCING CONTROL

The control for balancing the Pendubot is very similar to the classical cart-pole inverted pendulum problem. To design the controller we linearized the Pendubot's nonlinear equations of motion (3.9), and designed a full state feedback controller with the linear model. The Taylor series approximation

$$f_a(x,u) = f_a(x_r,u_r) + \frac{\partial f}{\partial x}\Big|_{x_r,u_r}(x-x_r) + \frac{\partial f}{\partial u}\Big|_{x_r,u_r}(u-u_r) \tag{6.1}$$

was used to linearize the plant. $\mathbf{x}$ is the vector of states given in equation (3.9). $\mathbf{u}$ is the single control input for the Pendubot. $\mathbf{x_r}$ and $\mathbf{u_r}$ are the equilibrium values of the states and control respectively. Since we are only interested in controlling the Pendubot at equilibrium positions, $f_a(\mathbf{x_r},\mathbf{u_r})$ will always be zero. All that is needed then is to find the partial derivative matrices and evaluate them at the equilibrium points. Studying equations (3.6) through (3.9) it is observed that the Pendubot's equilibrium points can be defined by

$$u_r = \theta_4 g \cos(x_{r1}), \tag{6.2}$$

$$x_{r1} + x_{r3} = \pi/2. \tag{6.3}$$

Differentiating equation (3.9) with respect to the states leaves the A matrix in the linearized model

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\partial f_2}{\partial x_1} & 0 & \frac{\partial f_2}{\partial x_3} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{\partial f_4}{\partial x_1} & 0 & \frac{\partial f_4}{\partial x_3} & 0 \end{bmatrix}. \tag{6.4}$$

The B matrix is found by the partial derivative with respect to the control input

$$\frac{\partial f}{\partial u} = \begin{bmatrix} 0 \\ \frac{\partial f_2}{\partial u} \\ 0 \\ \frac{\partial f_4}{\partial u} \end{bmatrix}. \tag{6.5}$$

Refer to Appendix B for a full derivation of these partial derivative terms.

Each equilibrium point defines a different linearized system (See Appendix B). This means that different control gains will be needed for each equilibrium point for best results in the balancing of the Pendubot. Most of our work with the Pendubot dealt with two

equilibrium points. We define the top balancing position as the upright position with $x_{r1}=\pi/2$, $x_{r3}=0$ and $u_r = 0$ (See Figure 2.2). The mid balancing position is defined as $x_{r1}=-\pi/2$, $x_{r3}=\pi$ and $u_r = 0$ (See Figure 5.2).

Using these equilibrium values and the parameters identified by the energy equation method (See section 4.2), the linear models for the top and mid equilibrium positions are as follows

Top

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 51.9265 & 0 & -13.9704 & 0 \\ 0 & 0 & 0 & 1 \\ -52.8402 & 0 & 68.4210 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 15.9549 \\ 0 \\ -29.3596 \end{bmatrix}$$

Mid

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -51.9265 & 0 & 13.9704 & 0 \\ 0 & 0 & 0 & 1 \\ 51.0128 & 0 & 40.4801 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 15.9549 \\ 0 \\ -2.5502 \end{bmatrix}.$$

Now given these linear models we can use LQR or pole placement techniques to design full state feedback controllers, u=-Kx. For example with

$$\mathbf{R} = [1]$$

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

the Matlab function "lqrd" can be used to derive optimal control gains for a discrete controller. Using a sampling period of 5ms the optimal gains are

$$\mathbf{K}_{Top} = [-32.68 \quad -7.14 \quad -32.76 \quad -4.88],$$

$$\mathbf{K}_{Mid} = [10.96 \quad 1.44 \quad 19.28 \quad 2.81].$$

The top and mid positions are not the only possible equilibrium positions of the Pendubot. In fact there is a continuum of equilibrium positions. Figure 6.1a shows another

possible configuration. There are four uncontrollable positions, $x_{r1}=0$, $x_{r3}=\pi/2$ or $-\pi/2$ and $x_{r1}=-\pi$, $x_{r3}=\pi/2$ or $-\pi/2$. Figure 6.1b shows the first of these positions.

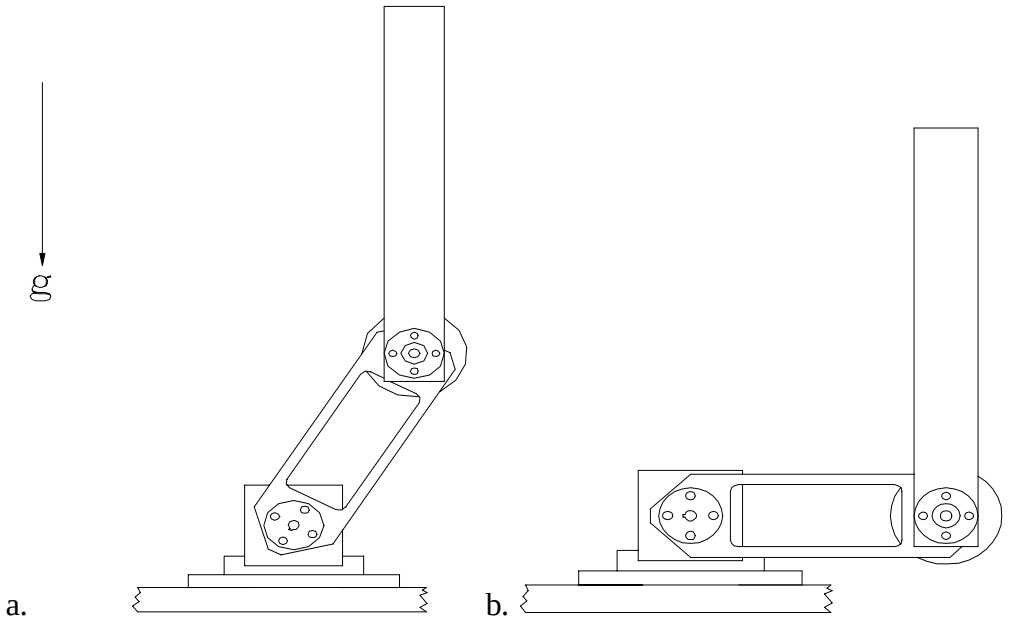

Figure 6.1 Other Possible Equilibrium Positions. a. Another controllable equilibrium. b. One of the four uncontrollable equilibrium.

In Chapter 9 response data is shown of a controller that demonstrates the capability of the Pendubot to balance in positions other than the top and mid positions. A gain scheduling technique is used to step the Pendubot to a new equilibrium position every 2 seconds. Each step moves the links closer to the uncontrollable equilibrium shown in Figure 6.1b, and in turn produces an equilibrium that is increasingly harder to control. Limited by the torque of the motor and the weight of the links, the Pendubot becomes unstable before it reaches the uncontrollable configurations. An exact study was not performed to see where the Pendubot became unstable, but approximately 40° offset from the top and mid positions was the limit for the motor balancing the links. See the source code “cirbot.c” listed in Appendix E for the gains used at each equilibrium position.

7. COMBINING AND IMPLEMENTING THE CONTROLLERS

With both the swing up control and the balancing control complete, an algorithm was needed to connect the two. Initially when working with only computer simulations of the Pendubot, the controllers were switched at a determined time (See Chapter 8 and Appendix D). This switch time was determined by observing when the swing up control had brought the links almost to rest at the desired equilibrium position. This worked very well for the simulation but behaved poorly when realized on the actual Pendubot. The reason being that the simulations are exactly repeatable but the actual runs are susceptible to different initial conditions and computational noise making them unable to repeat reliably. The following algorithm was used instead to give the Pendubot more intelligence and switch the control by watching the states of the system.

```
if   |x1- xr1| < .10 {
     if  |x3- xr3| < .20 {
        u = -Kx;
        if  |u| <  9 /* Volts */ {
           /* Output balancing control */
        } else /* Output swing up control */
     } else  /* Output swing up control */
  } else /* Output swing up control */
```

This algorithm waits for link one to arrive within 0.1 radians of its equilibrium position and then checks link two. If link two is also within 0.20 radians of its equilibrium position, the balancing control is calculated. If the control output is less then 9 volts, 10 volts being the maximum DAC output for the Pendubot, the control is switched to the balancing mode. Otherwise the links are passing too quickly though the equilibrium point and the swing up control remains intact.

Another implementation issue that arises in the controller design of the Pendubot is the approximation of the joint velocities. There is only position feedback in the system so a finite approximation is used to estimate the velocity. To find the velocity we simply used the finite difference method, [x(k)-x(k-1)]/sample period. This creates numerical error or noise in the calculation of the control effort, though, due to the finite resolution of the optical encoders. We found that simply taking the average of the last three velocities helped to filter

and decrease this noise. For example the velocity calculation for joint one, state $x_2(k)$, is written as follows

$$x_2(k) = \frac{x_1(k) - x_1(k-1)}{\Delta t}, \tag{7.1}$$

$$x_2(k) = \frac{x_2(k) + x_2(k-1) + x_2(k-2)}{3}. \tag{7.2}$$

A dither signal was also needed to help balance the links in the top position. Due to friction and the increased effect of gravity on the links in the top position, the balancing control was not able to hold the links motionless. The balance control at the mid position did not have this problem. The main reason for this is that gravity works with the control at the mid position to keep link one at the equilibrium. Where as with the top position, gravity works to pull both of the links away from the equilibrium. The motion produced was a large amplitude sway (approximately 0.2 radians). To eliminate some of this motion an open-loop dither signal was added to the control,

$$u = -\mathbf{Kx} + .25\sin 40.0t. \tag{7.3}$$

This dither signal helped to eliminate much of the sway but it was not able to cancel all of the motion. See Figure 9.1 for a plot of the motion around the equilibrium with the dither signal added.

8. SIMULATION RESULTS

This chapter displays the simulation results found when simulating the Pendubot with the software package Simnon [4]. The details of these simulations can be found in Appendix D which contains the Simnon program files used. Figure 8.1 shows a swing up, catch and balance of the Pendubot in the top position and Figure 8.2 shows the same for the mid position. This chapter serves as a comparison for the actual data shown in the following chapter.

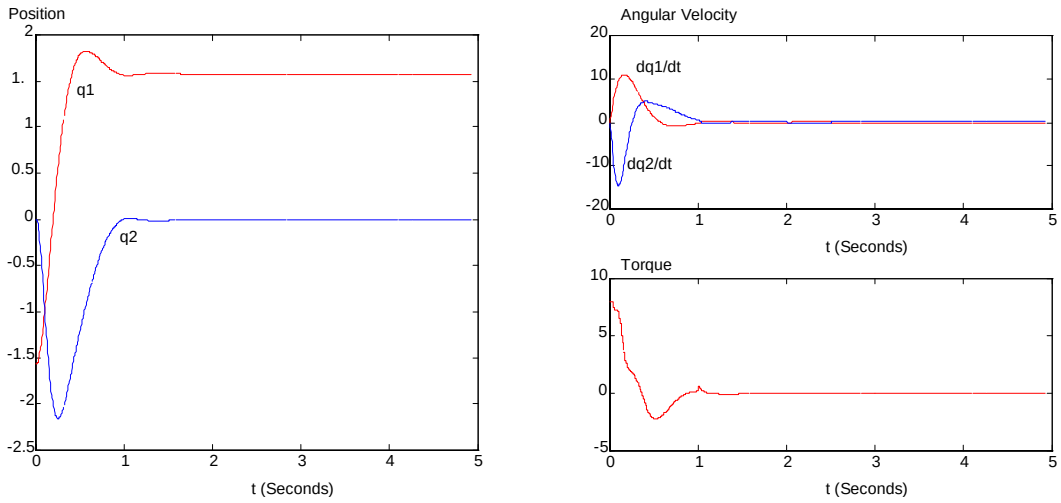

Figure 8.1 Simulation in Simnon: Swing Up to the Top Position.

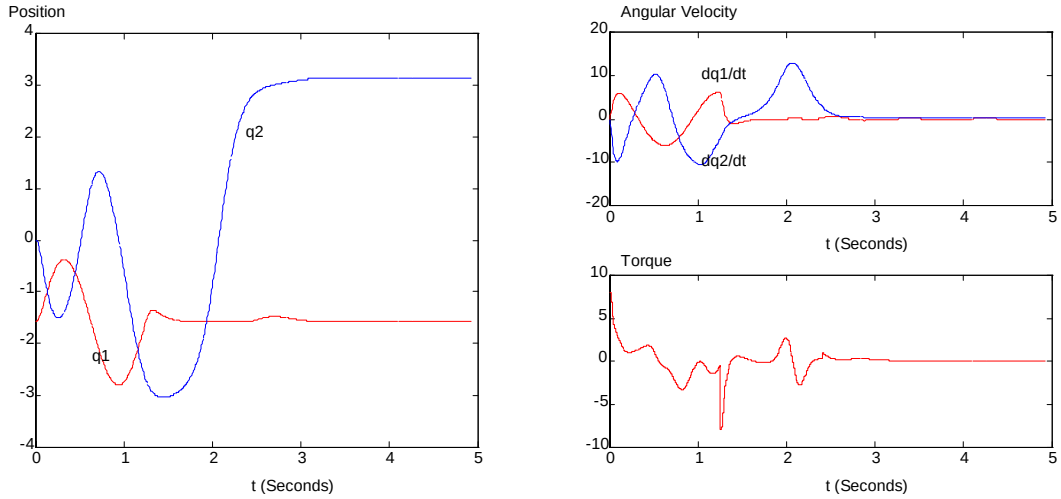

Figure 8.2 Simulation in Simnon: Swing Up to the Mid Position.

9. EXPERIMENTAL RESULTS

This final chapter shows actual responses of the Pendubot system. The balancing controller gains for these runs were found by pole placement methods and are listed in the figure's description. Figures 9.1 and 9.2 show the Pendubot swinging and balancing at the top and mid positions respectively. Figure 9.3 demonstrates the ability of the Pendubot to balance at its many equilibrium points. The control used for Figure 9.3 swings the links to the mid position and then steps the links at 5° increments away from the mid position every two seconds. Each new equilibrium point has its own balancing control gains and an equilibrium control $u_r = \Theta_4 g\cos(x_{r1})$ which is no longer zero.

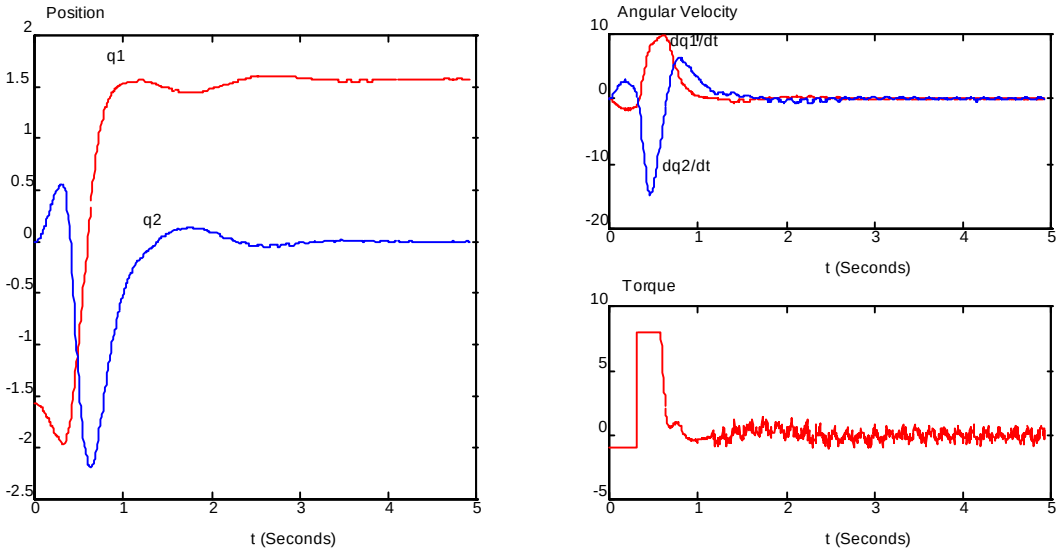

Figure 9.1 Swing Up and Balance Control at the Top Position. Outer Loop control gains used for the swing up control: Kp=150.0,Kd=21.7. Full state feedback gains used for the balancing control: K=[-27.48 -6.07 -28.58 -4.24].

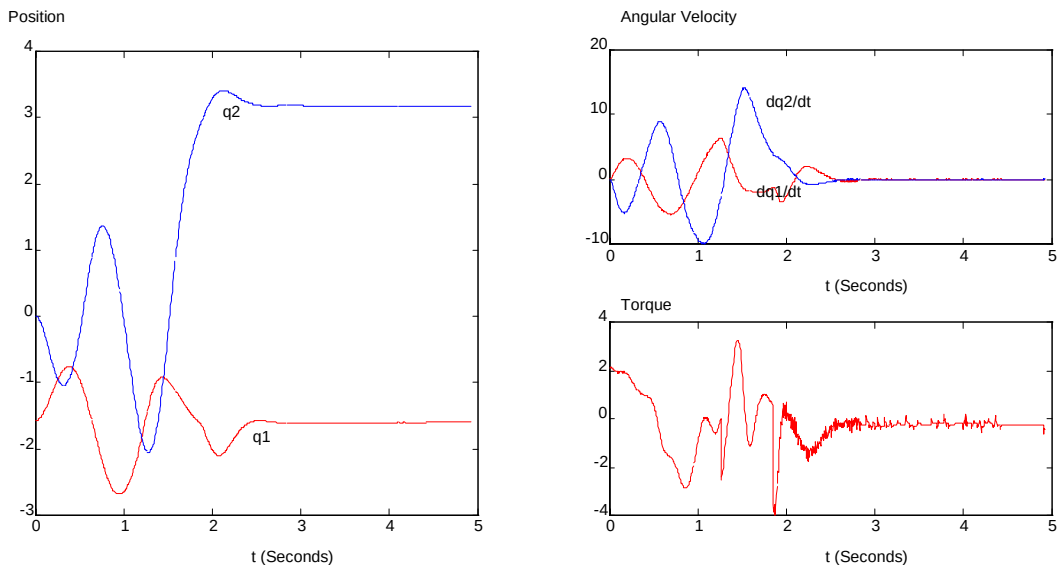

Figure 9.2 Swing Up and Balance Control at the Mid Position. Outer Loop control gains used for the swing up control: Kp=32.0,Kd=4.85. Full state feedback gains used for the balancing control: K=[15.31 1.76 22.86 3.38]. Swing up trajectory: 1.4sin(5t) - π/2.

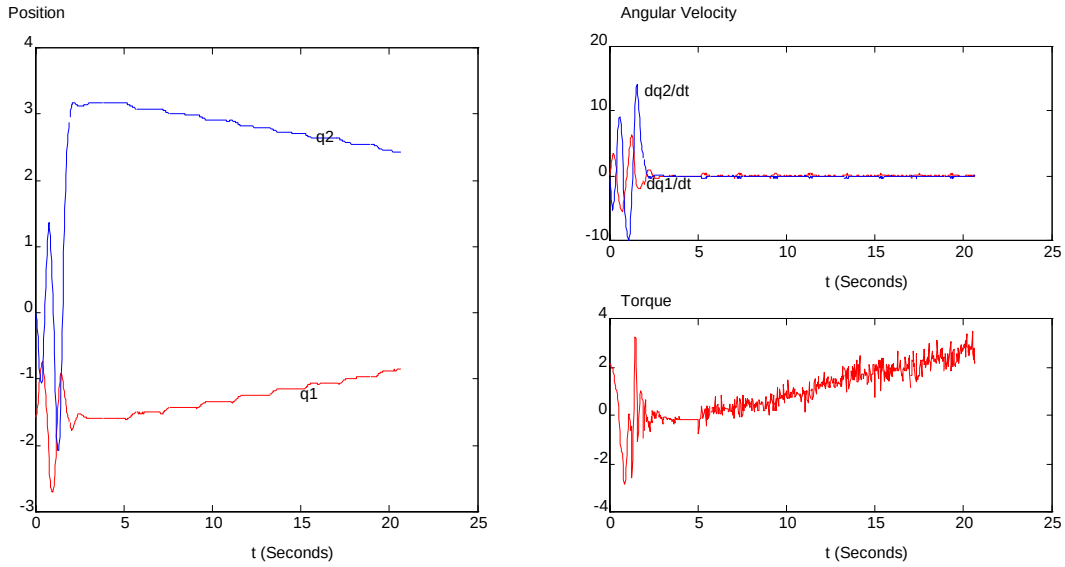

Figure 9.3 Demonstration of other Balancing Positions. This plot first shows the swing up and balance control at the mid position identical to Figure 9.2. This plot also goes further and demonstrates the Pendubot's balancing capabilities at other equilibrium points. Full state feedback gains change for each equilibrium.

10. CONCLUSION

This thesis presented our new design of a two link underactuated planar revolute robot, named the Pendubot. Its actuated joint is located at the shoulder and the elbow joint is unactuated and allowed to swing free. The controller for the Pendubot was implemented using data acquisition cards and an IBM compatible 486DX2 PC. Three different parameter identification methods were used to identify the unknown dynamic parameters of the linkage. First the parameters were found off-line by creating a solid model of the Pendubot in a CAD package. The second method is an on-line method that takes advantage of the energy equations of the linkage which are linear in terms of the unknown dynamic parameters. A simple least squares problem was then derived to solve for the parameters. The third method, also on-line, uses a constrained minimization algorithm to minimize the error between actual collected response data and simulated response data. Two controllers were designed for the Pendubot. Partial feedback linearization techniques were used to design the control that swung the links from their hanging stable position to unstable equilibrium positions. Then to catch and balance the second link at the unstable equilibrium, full state feedback controllers were designed using the linearized model of the links at the desired position. Results were presented demonstrating the performance of the Pendubot with these controllers.

REFERENCES

[1] Bortoff, S.A., Pseudolinearization using Spline Functions with Application to the Acrobot, Ph. D. Thesis, Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 1992.

[2] Spong, M.W., "The Control of Underactuated Mechanical Systems", Plenary lecture at the First International Conference on Mechatronics, Mexico City, January 26-29, 1994.

[3] Spong, M.W., "Swing Up Control of the Acrobot using Partial Feedback Linearization," SY-ROCO'94, Capri, Italy, pp. 833-838, September, 1994.

[4] Elmquist, H., *Simnon-User's Guide*, Dept. of Automatic Control, Lund Inst. of Tech., CODEN:LUTFD@/(TFRT-3091), 1975.

[5] Spong, M.W., and Vidyasagar, M., *Robot Dynamics and Control*, John Wiley & Sons, Inc., New York, 1989.

[6] Gautier, M., and Khalil, W., "On the Identification of the Inertial Parameters of Robots. In *Proceedings of 27th CDC*, pp. 2264-2269, 1988.

[7] Prüfer, M., Schmidt, C., Wahl, F., "Identification of Robot Dynamics with Differential and Integral Models: A Comparison", Proc. IEEE Int. Conf. on Robotics and Automation, San Diego, CA, pp. 340-345, May, 1994.

[8] Chipperfield, A.J., and Fleming, P.J., *MATLAB Toolboxes and Applications for Control*, Peter Peregrinus Ltd, Stevenage, UK, 1993.

[9] Papalambros, P.Y., and Wilde, D.J., *Principles of Optimal Design, Modeling and Computation*, Cambridge University Press, New York, 1991.

[10] Spong, M. W., and Block, D. J., "The Pendubot: A Mechatronic System for Control Research and Education, "*34th IEEE Conf. on Decision and Control*", New Orleans, Dec., 1995, submitted.

[11] Block, D. J., and Spong, M. W., "Mechanical Design and Control of the Pendubot, "*SAE Earthmoving Industry Conference*", Peoria, IL, April 4-5, 1995.