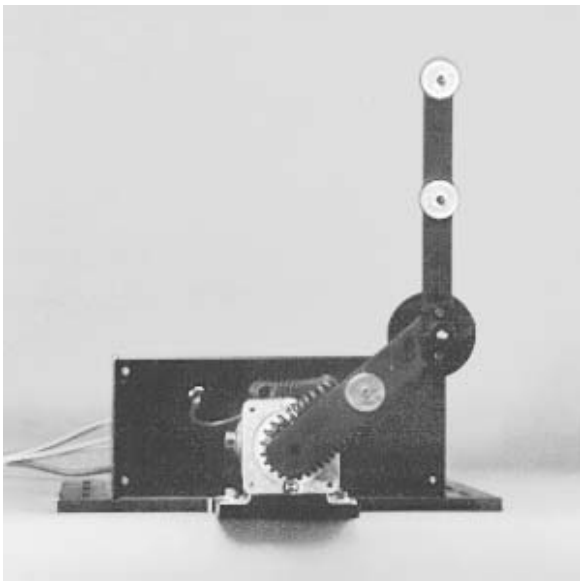
PENDUBOT
USER'S MANUAL



Mechatronic Systems, Inc.
P.O. Box 11196
Champaign, IL 61826-1196
Telephone: (217) 355-5396
Fax: (217) 244-1653
Web Page: http://www.prairienet.org/msi

IMPORTANT NOTICE!

Do not attempt to operate the Pendubot without proper supervision and without first reading carefully the contents of this User's Manual! Pay strict attention to the safety precautions in Section 2.3! Mechatronic Systems, Inc. assumes no responsibility for injuries caused by improper installation or improper use of either the hardware or software!

Contents

Chapter 1

Introduction

Thank you for purchasing the **Pendubot Model P-2**, inverted pendulum control experiment from **Mechatronic Systems, Incorporated.** The Pendubot is the latest innovation for instruction and research in real-time control and identification. The Pendubot may be used at all levels of instruction in control and robotics:

- in **freshman level** courses as a demonstration tool to motivate important concepts in dynamics and control, and the systems approach to engineering design,
- in **junior level** courses in frequency domain and state space control system analysis and design,
- in **senior level** courses in digital control, mechatronics, and real time programming,
- and in **graduate level** courses in linear and nonlinear control, intelligent control, and robotics.

The Pendubot is also sophisticated enough to be used as a **research tool** to investigate :

- geometric nonlinear control
- robust and adaptive control and identification
- intelligent control, including neural networks, fuzzy logic, and genetic algorithms
- hybrid and switching control.
- modeling, identification and control of friction
- analysis and control of chaotic dynamics.

Before attempting to install and operate the Pendubot it is important that you familiarize yourself with all hardware and software supplied. Read all sections of this manual completely. **Pay particular attention to the Safety Precautions in Section 2.3**. In addition you should read through the manuals included from the third party vendors. The *MSI Getting Started Disk* also includes a readme file (README.1ST). You should fully understand the contents of that disk as well before attempting to operate the Pendubot.

1.1 About Mechatronic Systems, Inc.

Mechatronic Systems, Inc., was founded in 1996 and is dedicated to the manufacture and sale of electro-mechanical devices for use in control engineering research and education. We offer high quality, custom built devices at competitive prices.

1.2 How to Contact Us

FOR SALES AND SERVICE:

Mechatronic Systems, Inc.
P.O. Box 11196
Champaign, IL 61826-1196
phone: (217) 355-5396
fax: (217) 244-1653
email: msi@prairienet.org

FOR TECHNICAL SUPPORT:

Daniel Block
phone: (217) 244-8573
fax: (217) 333-7427
email: d-block@uiuc.edu

WORLD WIDE WEB:

http://www.prairienet.org/msi/

1.3 What is the Pendubot?

The **PENDUBOT**, short for **PENDU**lum Ro**BOT**, is an electro-mechanical (or **Mechatronic**) system consisting of two rigid links interconnected by revolute joints as shown in Figure (1.1).

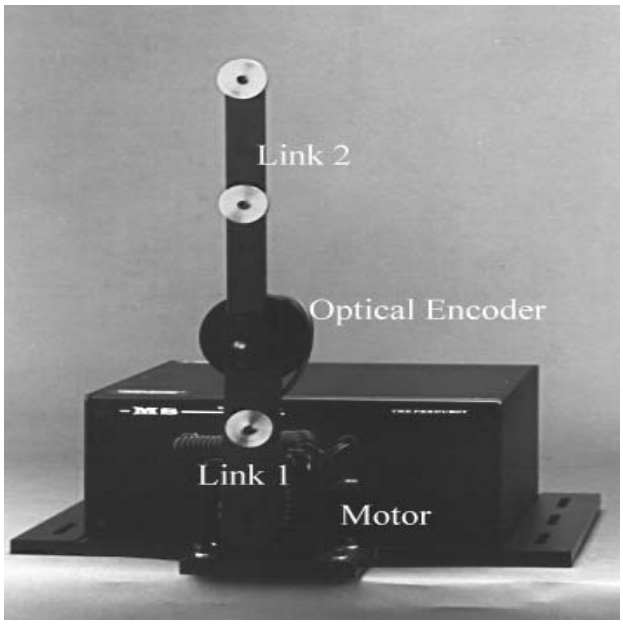


Figure 1.1: The Pendubot

The first joint is actuated by a DC-motor and the second joint is unactuated. Thus the second link may be thought of as a simple pendulum whose motion can be controlled by actuation of the first link. The Pendubot is similar in spirit to the classical inverted pendulum on a cart. However, the nature of the dynamic coupling between the two links of the Pendubot results in some interesting properties not found in these other devices. We will discuss these properties further in Section 4.1.

1.4 Description of the Hardware

The Pendubot consists of two rigid aluminum links machined from $\frac{1}{4}$–inch (0.635–cm) thick aluminum. Link 1 is 6 inches (15.24–cm) long and is directly coupled to the shaft of a 90V permanent magnet DC motor mounted on the supporting base. Link 1 also includes the bearing housing for joint two. Needle roller bearings riding on a ground shaft are used to construct the revolute joint for link 2. The shaft extends out both directions of the housing allowing coupling to the second link and to an optical encoder mounted on link 1. The design gives both links full 360^o of rotational motion. Link 2 is 9 inches (22.86–cm) long and contains a coupling that attaches to the shaft of joint two.

Two *Dynamics Research Corporation* 1250 counts/rev resolution optical encoders provide position feedback, one attached at the elbow joint and one attached to the motor. An *Advanced Motion Controls*

25A8 PWM servo amplifier is used to drive the motor. The amplifier is used in torque mode and adjusted for a gain of 1V=1.2Amps.

All of the control computations are performed on an IBM compatible PC-workstation [1] with a D/A card, an encoder interface card, and a timer card installed in the PC. A CIO-DAC-02 card from *Computer Boards, Inc.* is used for digital to analog conversion and a *Dynamics Research Corporation* optical encoder card is used to interface with the optical encoders. Controller timing is provided by a timer board from *Computer Boards, Inc.* that utilizes the 9513 timer chip. Using the standard software library routines supplied with the interface cards together with our own drivers we are able to program control algorithms directly in C. See the Appendix for the addresses of the above vendors.

1.5 What Can You Do With the Pendubot?

The Pendubot possess many attractive features for control research and education. It's dynamic properties will be discuss in Section 4.1. Using the Pendubot one can investigate system identification, linear control, nonlinear control, optimal control, learning control, robust and adaptive control, fuzzy logic control, intelligent control, hybrid and switching control, gain scheduling, and other control paradigms. One can program the Pendubot for swingup control, balancing, regulation and tracking, identification, gain scheduling, disturbance rejection, and friction compensation to name just a few of the applications.

Below is a description of several interesting properties and problems associated with the Pendubot that you can use to develop control projects and to design and implementing your own algorithms.

1. **Identification:** The first step in any control system design is to develop a mathematical model of the system to be controlled. The Pendubot dynamics are described by coupled nonlinear second order differential equations as shown in Section 4.1. These equations are most easily derived using Lagrangian mechanics [12]. These equations, although nonlinear in terms of the joint variables, are *linearly parametrizable*. This means that the inertia parameters, such as the link masses, moments of inertia, etc. can be grouped in such a way to define a set of parameters that appear linearly in the equations of motion (see Section 4.1). This makes it possible to perform on-line parameter identification using the method of least squares [2].

2. **Balancing:** Once the parameters describing the Pendubot dynamics are identified, the problem of balancing the Pendubot about an open loop unstable equilibrium may be investigated. The Pendubot possesses infinitely many configurations at which it can be balanced. This "equilibrium manifold" is characterized in Section 4.2. On this manifold, the Taylor series approximation results in a controllable linear system[2]. We have supplied controllers design using Linear Quadratic Optimal Control theory to balance the Pendubot at two of these configurations, namely, $q_1 = \pi/2$, $q_2 = 0$ and $q_1 = -\pi/2$, $q_2 = \pi$. You may design additional such controllers for these and other equilibrium configurations using other methods such as fuzzy control, pseudolinearization, robust control, variable structure control, etc.

3. **Swingup:** The problem of swinging the Pendubot from the open loop stable configuration $q_1 = -\pi/2$, $q_2 = 0$ to any of the inverted equilibria is an interesting problem because of the strong

[1] We recommend a Pentium 100MHz or higher running Windows 95.

[2] Actually, there are four configurations which are linearly uncontrollable, $q_1 = 0, \pi$, $q_2 = \pm\pi/2$.

nonlinearity and dynamic coupling between the links. We have supplied controllers that are based on the notion of partial feedback linearization [6] and nonlinear zero dynamics that can be used for swingup control. You may design additional controllers using passivity or energy methods, heuristic, fuzzy logic, or machine learning methods, or a host of other available techniques.

4. **Swingup and Balance:** Combining the operations of swingup and balance is an ideal problem to investigate so-called hybrid and switching controllers. So-called *Logic Based Switching* control is a relatively new approach to designing robust controllers for complex systems such as the Pendubot.

5. **Friction Compensation:** The effect of friction in the motor brushes and bearings generally results in limit cycle behavior unless actively compensated by the controller. Friction modeling and control is of current research interest and the Pendubot is a good vehicle for experimental verification of theoretical results in this area. The controllers supplied by us include a small dither signal that reduces the amplitude of the limit cycle.

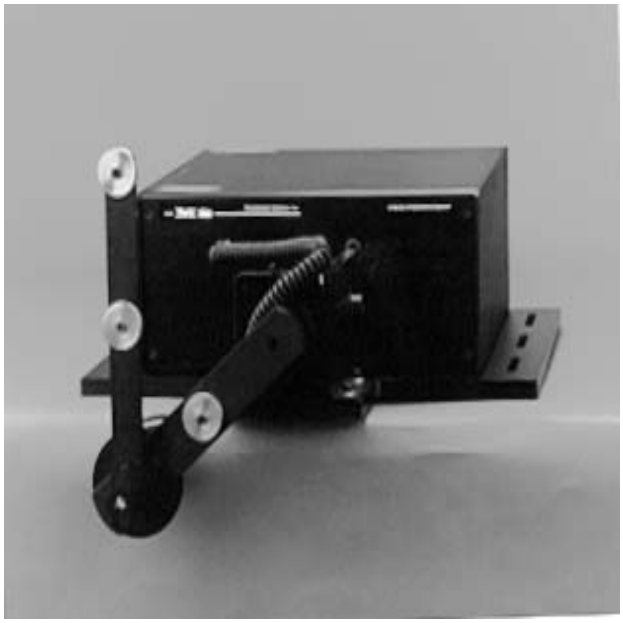


Figure 1.2: The Pendubot balancing along its equilibrium manifold

Chapter 2

Setting Up and Testing the Pendubot

The next two chapters describe how to set up the Pendubot on your own workbench, install and test the software, run the controllers supplied with the Pendubot, and write your own controller software. This chapter describes the technical details of getting the Pendubot up and running with your PC. The chapter is not intended to be all inclusive in teaching you how to use the Pendubot and its components. For this reason it is important that you take the time to study the hardware and software that are supplied before you try out the swing up and balancing controllers.

Be sure to read and understand completely this manual and the manuals supplied by the third party vendors before attempting to operate the Pendubot.

2.1 What is Included

You should check carefully to determine if all of the components listed below have been received in working order.

1. The Pendubot base unit consisting of the mounting plate, case, power supply, amplifier, and all necessary connectors.
2. Mechanical linkage with attached optical encoder and cable connector,
3. Computer Boards, Inc., CIO-DAC-02 Digital to Analog Output PC card, software and manual,
4. Computer Boards, Inc., PC-timer card CIO-CTR05.
5. Dynamic Research Corp. encoder interface PC card, software, and manual (The manual is on the DRC disk).
6. Cable for DAC-02 and DRC encoder interface

7. MSI *Getting Started Disk*. Refer to Section 2.5 *Installing the Software* for a description of the software supplied with the Pendubot.

2.2 What You Will Need

Along with the Pendubot and the various boards that are supplied, you will need:

1. A Pentium PC (100 MHz or higher) running Windows95, Windows98, or WindowsNT 4.0 with at least three available ISA slots.
2. A C-compiler. For the 16 bit software, Microsoft Visual C++, Version 1.5.2 is recommended. For the 32 bit software, Microsoft Visual C++, Version 6.0 is recommended. The 16 bit and 32 bit software is described more fully later.
3. A sturdy table or workbench. It is highly recommended that you secure the Pendubot base plate to the table top with bolts or screws using the mounting slots provided on the base plate in order to avoid sliding of the base plate during operation. The mounting holes are drilled for 1/4-inch (6.35-mm) bolts. You will have to supply your own mounting bolts or screws.
4. An Oscilloscope or Digital Volt Meter (DVM) is also recommended for monitoring the control output from the DAC. We recommend that you monitor the DAC output at all times when running the Pendubot. That way you can monitor your controller's output and catch intermittent errors in your newly developed controllers.

2.3 Safety Precautions

As with all robotic devices, care must be taken when working with the Pendubot to avoid injury. The motor is capable of moving the aluminum links at high velocities. Even with no power to the motor, the free response of the Pendubot can be hazardous. The free response is approximately that of a double pendulum which is known to possess chaotic solutions. If the Pendubot is held at a position far away from its stable equilibrium and allowed to swing freely, the motion can be quite violent. This not only presents a risk of injury, but can also result in damage to the system from cable windup and possibly damage to the bearings and shafts.

THE FOLLOWING SAFETY PRECAUTIONS SHOULD BE OBSERVED AT ALL TIMES:

1. The amplifier inhibit switch (or deadman switch) supplied with the Pendubot should never be bypassed. Even with the deadman switch in place, considerable care should be taken when implementing control algorithms, as the Pendubot may swing violently when power is disengaged.
2. Upon completion of any control experiment to balance the Pendubot, it is recommended that you continue to depress the deadman switch and firmly grasp the Pendubot before releasing the deadman switch. After releasing the deadman switch, slowly lower the Pendubot to its lower or

stable equilibrium configuration. **Do not allow the Pendubot to fall freely from its inverted position. Also, do not attempt to grasp the Pendubot if it is rotating rapidly as this could result in personal injury.**

3. Never place your hands or your head within the workspace of the Pendubot while it is operating (even while it is balancing) and be careful when touching the Pendubot while it is operating. Be aware that long hair or loose clothing may become entangled in the mechanism and take appropriate precautions.

4. To avoid the risk of electrical shock, do not operate the Pendubot with the metal covers off and do not touch the power amplifier while power is on.

5. Do not open the PC, or install or remove the boards without first turning off all power. Use a grounding wrist strap to avoid buildup of static electricity while installing or removing any of the boards.

2.4 Mounting the Pendubot Base

It is possible to operate the Pendubot without attaching it to the worksurface. However, the swinging motion of the Pendubot links may cause the base to move. For this reason it is strongly recommended that you attach the Pendubot securely to a table or workbench prior to operation. The base plate of the Pendubot contains mounting holes, which are drilled for 1/4-inch (6.35-mm) bolts (not supplied) as shown in the accompanying drawing on the next page. You should place the Pendubot in a secure location so that the Pendubot can swing freely without hitting any obstacles. Mark the locations of the mounting holes on the table, drill the holes, and secure the Pendubot to the table with four (4) suitable bolts.

****

CAUTION!

You must hang the base plate far enough off the edge of the table (approximately 5 cm) so that the encoder on link 1 does not hit the table when it swings.

****

2.5 Installing the Software

We have supplied you with three different versions of the software. These are in the form of self extracting, executable files. On the disk labeled MSI Getting Started Disk - 16 bit Software are the files `MSIWIN31.EXE` and `MSIWIN95.EXE`. On the disk labeled MSI Getting Started Disk - 32 bit Software is the file `MSIWIN32.EXE`. Both disks contain, in addition, a `README.1ST` file which provides additional information.

The version of the software that you should install depends on which version of Microsoft Windows you are running and which version of Matlab you have. Both `MSIWIN31.EXE` and `MSIWIN95.EXE` contain 16-bit code while `MSIWIN32.EXE` contains the 32-bit software. Refer to the table below to determine which version to install.

Operating System	Matlab version	MSI executable
Windows 3.1	Matlab 4.2	MSIWIN31.EXE
Windows 95	Matlab 4.2	MSIWIN95.EXE
Windows 95,98,NT4.0	Matlab 5.0 or higher	MSIWIN32.EXE

If you are installing the software on a machine that is running only DOS, you may install either `MSIWIN31.EXE` or `MSIWIN95.EXE`. The only difference between these two files are the PIF (program interface files) which are not used by DOS. In addition, both MSI Getting Started disks contain a `README.1ST` file which provides additional information.

To install MSIWIN31.EXE or MSIWIN95.EXE follow the steps below. To install MSIWIN32.EXE, follow the same steps but replace the directory name "C:\MSI" with "C:\MSIWIN32".

1. Create a directory called "C:\MSI" (at the DOS prompt type "MD MSI")

   See the Note ** below if you do not want to install the software in the directory C:\MSI.

2. Change to the C:\MSI directory (type "CD C:\MSI")

3. Copy the file README.1ST and the appropriate executable file, MSIWIN95.EXE, MSIWIN31.EXE, or MSIWIN32.EXE into C:\MSI.

4. To extract the files, type "MSIWIN95 -D" or "MSIWIN31 -D" or "MSIWIN32 -D" depending on which executable you wish to run and the files will be extracted to the correct directories.

5. Both the DOS controllers and the Windows executable assume that Matlab is installed on your computer in the directory "C:\MATLAB". If this is not the case, the create the directory "C:\MATLAB" and put that directory in your MATLABRC.M file if you are running Matlab 4.2 or else put "C:\MATLAB" in your Matlab path if you are running Matlab 5.0 or higher by using the path editing tool.

6. You may now want to create a Program Group and Program Item (in Windows 3.1) or a Short-cut (in Windows 95,98,NT) for the MSDEV.EXE/MSDEV32.EXE executable supplied. See your Windows documentation if you are not sure how to do this.

7. **For Windows NT only:** You must install the device driver supplied in the "NTDRIVER" directory. Please refer to the `readme.txt` file in this directory for further instructions.

The Installation is now complete.

Note **If for some reason you already have a directory called C:\MSI (or C:\MSIWIN32) on your computer, you can install the software in another directory name that you choose. If you do this you will need to change the default directory in the Windows executable code, MSDEV.EXE (or MSDEV32.EXE if you are using the 32-bit software). To change it permanently edit the file MSDEVDOC.CPP and in the initialization function "OnNewDocument" change the initialization of the member variable `m_msdev_path` to the new path where MSDEV.EXE/MSDEV32.EXE is located. For the 16 bit software you should also edit the PIF file for the controller's executable file to indicate the correct path.

2.6 Installing the PC Cards

To setup and install the two interface cards,CIO-DAC02,CIO-CTR05, from *Computer Boards* and the single encoder interface board, from *DRC*, you will need to read the instructions supplied with each board. Each board should be setup with the following settings:

Note!!! we have already preset these boards for you before shipping but you should double check them

CIO-DAC-02:Hex Address0x300
Wait State:ON
DAC 0 Voltage Reference:-10V
DAC 1 Voltage Reference:-10V

CIO-CTR-05:Hex Address0x310
Wait State:ON
Interrupt Level: NONE

DRC Encoder Interface:
Encoder 1: Hex Address0x210

Encoder 2: Hex Address0x220

If other devices in your computer conflict with these addresses you can of course choose your own address settings for each card, but you will need to change and then recompile the source code given in order for the supplied controllers to work. For the two ComputerBoard cards (CIO-DAC-02 and CIO-CTR-05), the calibration executable "InstaCal" that is supplied with their installation disk has a install routine that produces a picture of the board indicating how your jumpers and dip switches should be set. For the optical encoder board the supplied disk contains a file named "pcmanual.doc". Read through this document for setup instructions and other aspects of the encoder card. Once you have all the jumper settings correct on the boards, install them in your PC in three ISA slots.

For future reference note that the two ports (9-pin D-SUB connectors) on the DRC card are labelled **P1** and **P2**. Port **P1** is closest to the mounting screw on the card and will be connected to the cable labeled **E1**. Likewise port **P2** will be connected to the cable labeled **E2**. The **E1** and **E2** connectors are on the flat ribbon cable described in the next section. Note which 9-pin connector corresponds to these ports **P1** and **P2** before installing the DRC cards.

2.7 Testing the Components

This section discusses how to connect the PC cards to the Pendubot and test them.

****

CAUTION!

Turn off all power to the computer and the amplifier when connecting and disconnecting cables and wires. Do not plug and unplug the optical encoders from the optical encoders interface board when the PC is on. Make sure the PC is off when connecting and disconnecting the encoders.

****

2.7.1 Connecting the Pendubot to the Computer

1. Turn off the PC and the Pendubot main switch.
2. Locate the flat ribbon cable supplied with the Pendubot. Connect the 37-pin connector (the largest connector) of the flat ribbon cable to the back of the Pendubot.
3. The other end of the flat ribbon cable has three connectors labeled D1, E1, and E2. Assuming that you have already installed both the DRC card and the DAC-02 card in your PC, connect E1 and E2 to the ports P1 and P2, respectively, on the DRC card and connect D1 to the 25-pin connector on the DAC-02 card.
4. Connect the Amplifier Inhibit switch to the phone jack on the back of the Pendubot.
5. Connect the Pendubot power cord but do not turn on the Pendubot power at this time. In fact, the Pendubot main power switch should never be turned on until after the PC has finished booting and that you are ready to run a controller.
6. Turn on the PC but leave the Pendubot main power switch off. Edit your AUTOEXEC.BAT file and insert the line

   C:\MSI\CNTRLERS\ZERODAC\ZERODAC

   for the 16 bit software or

   C:\MSIWIN32\CNTRLERS\ZERODAC\RELEASE\ZERODAC.EXE

for the 32 bit software.

Note: If you are using this software under WindowsNT 4.0 you will have to add ZERODAC.EXE to the startup folder for each user. In order for this to work properly the NT device driver should be setup to start automatically.

The executable file ZERODAC.EXE has already been copied to the correct directory if you have followed the procedure to unpack and install the software in Section 2.5. The program ZERODAC.EXE will be executed each time you restart your computer and will ensure that the DAC output is set to zero. Never turn on the Pendubot main power switch or press the amp inhibit switch before the DAC output has been zeroed, i.e., before the computer has finished booting.

7. Reboot your computer.

2.7.2 Testing the Encoder Interface Board

The DRC encoder interface cards are supplied with demo programs that should now be executed in order to determine if the encoder cards are communicating properly with the encoders and if they are counting correctly. For operating systems other than WindowsNT 4.0, you should run the demo program called `DRC_CARD.EXE` . If you are running under WindowsNT 4.0, this program will not work. Instead you should run the program `TESTENC.EXE` located in the "CNTRLERS" directory.

Facing the Pendubot, rotate the motor shaft by hand and observe whether the monitor display for encoder 1 changes. If you are running `DRC_CARD.EXE` , counterclockwise rotation of the motor shaft should count negative. If you are running `TESTENC.EXE` , instead, then counterclockwise rotation of the motor shaft should count positive.

Note: We define the positive rotation for the Pendubot linkage to be counterclockwise. Our include file, ENCODRBD.H, automatically accounts for the correct sign convention for the encoder counts.

You will not be able to test encoder 2 until after you mount the Pendubot linkage to the motor shaft. It is a good idea to test encoder 2 later, when you attach the Pendubot linkage, using this same procedure.

2.7.3 Testing the DAC-02 Digital/Analog Interface Board

To test the CIO-DAC-02 card, read its output voltage with a Digital Voltmeter (DVM). You may do this by connecting the DVM to DAC plus and minus ports on the back of the Pendubot enclosure. Use the program

"C:\MSI\CNTRLERS\DAC02\DAC02.EXE"

for the 16 bit software, and

"C:\MSIWIN32\CNTRLERS\DAC02\RELEASE\DAC02.EXE"

for the 32 bit software, to change the voltage output and check that the desire output and actual output match.

2.7.4 Testing the CTR-05 Timer Board

To test the CIO-CTR-05 use the program

"C:\MSI\CNTRLERS\CTR05\CTR05.EXE"

for the 16 bit software, and

"C:\MSIWIN32\CNTRLERS\CTR05\RELEASE\CTR05.EXE"

for the 32 bit software. This executable commands the timers to start counting. Counter 1 at 10ms periods, Counter 2 at 20ms periods, and Counter 3 at 40ms periods. You will have to connect an Oscilloscope to the counter's output pin to verify that it is counting. See the CIO-CTR-05 users manual for pin locations for the timer outputs and digital ground.

To test the motor and amplifier, make sure the Pendubot links are removed from the motor shaft. Turn on the Pendubot main power switch on the back of the Pendubot base unit. Then again use the DAC02.EXE program to send a small voltage (0.1V to .2V) to the amplifier. When you depress the amplifier inhibit switch you should see the motor spin. Try negative and positive voltages to make sure the motor spins in both directions.

2.8 Attaching the Pendubot Links

To mount the Pendubot linkage slide the upper bracket of link one onto the motor's shaft. Make sure that the flat of the shaft is oriented so that two of the set screws are perpendicular to the flat's surface. Also make sure that you slide the bracket close to the motor's face but of course not touching it. Try to leave about .5 to 1 mm spacing. You will then need to tighten the **four** 10-32 set screws using a 3/32 Allen key. Tighten the screws securely but make sure not to stripe the tapped hole.

Attach the cable from the encoder on the Pendubot link 2 to the front of the Pendubot base unit. Using one of the plastic tie wraps supplied, secure the encoder cable to the motor front plate. Leave enough slack so that the link may rotate freely. Refer to the Figures on the following page as a guide.

2.9 Testing the Pendubot

We have set up the amplifier so that it is in current mode and has a gain of 1V in = 1.2Amps out. These settings are set by the dip switches and potentiometers on the amplifier. If you adjust any of these, the parameters of your system will change and you will have to make appropriate adjustments. You can check the gain of the amplifier by applying a small constant voltage to the amplifier (no greater than 1.25 volts so that the links do not start spinning out of control if the link slips out of your hand) and then while holding link one press the inhibit button and read the current applied. To read the applied current use a DMM to read the voltage at the current monitor banana jacks on the back of the Pendubot's enclosure. Multiply this voltage by 4 (4A=1V) to find the applied current. See Advanced Motion Control's documentation for more details.

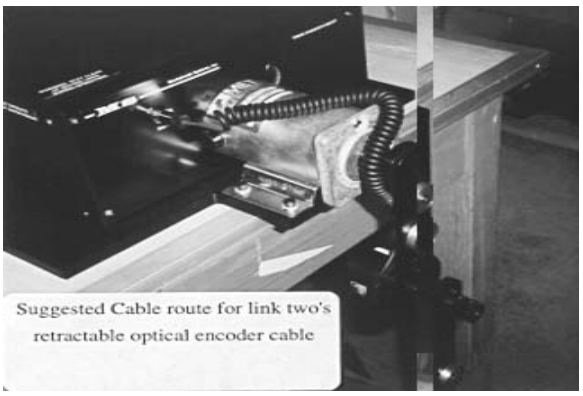


Figure 2.1: Suggested Position for Optical Encoder Cable

Figure 2.2: Attachment Point for Encoder Cable

Chapter 3

Running the Control Software

We have included several controllers for swing up and balance control of the Pendubot. In addition, we have supplied a graphical user interface (GUI), named MSDEV.EXE (or MSDEV32.EXE if you have installed the 32 bit software), to simplify the task of running controllers, changing gains, plotting results, etc. This chapter explains how to run the controlling software and how to write your own controllers. You should refer to the chapter that follows for the theory behind the dynamics of the Pendubot and the design of controllers for the Pendubot.

In Section 3.1 we discuss the 16 bit software for use under DOS and Windows 3.1/95/98. Users who install the 32 bit software for use with Windows 95/98 or Windows NT 4.0 should skip to Section 3.5. Additional information not included in the manual can be found in the readme files on the installation disks.

3.1 Overview of the 16 bit Software Provided

The 16 bit controller software that we have provided, MSIWIN31.EXE and MSIWIN95.EXE, is written to run in the DOS environment. The controllers can run in both plain DOS and in DOS boxes under Windows 3.1 and Windows 95/98. To run in a DOS box, a PIF file must be created to instruct Windows in how to execute the program. We discuss PIF files in the section below entitled *Running Controllers in Windows*.

We have supplied you with eight controllers. Six of them are very similar, differing in only the user input. These controllers swing up and balance the Pendubot in the "mid" and "top" operating points. The last two of the controllers implement gain scheduling, controlling the Pendubot along sinusoidal trajectories. We have included all the source code for these controllers, so once you become familiar with the Pendubot you can design your own control algorithms for the system.

The documentation for the controllers is found in each controller's source code. We feel this is a better way for you to learn what our controllers are doing instead of trying to reproduce the code here in this manual. So before you run a controller you will need to study the source code provided. The eight controller executables can be run, like any other DOS program, by typing the executables name at the

DOS prompt and if desired also type parameter inputs for the controller.

For example to run the BALMID.EXE executable, type at the DOS prompt: "BALMID DATA.M 150 21 4.5 1.65". BALMID will then run with the given input parameters. If you do not type parameter values or type the wrong number of parameters the default set of parameters are used.

The Graphical User Interface

We have also supplied you with a Windows GUI (or Graphical User Interface) for running the DOS controllers. When you are first developing your own controller you probably will want to run out of a DOS session. But in a classroom setting or when you are running a number of controllers for demos or experimentation you may find this Windows interface handy. If you have installed the software using MSIWIN31.EXE or MSIWIN95.EXE then the windows GUI program is called "MSDEV.EXE". This program allows you to save control files (.CTL) that keep track of the gains or other parameters sent to the controller executable and also interface to the appropriate version of Matlab to plot your response data. More details can be found in the help file entitled README.WIN. We have supplied you with the entire source code for the GUI programs.

The Controller Source Code

The next thing to do is get familiar with the controller's source code (BALMID.C, BALTOP.C, etc.). Study the code so you understand what the controller is doing. Once you feel confident you understand the system you can try the controller programs we supplied. The swing up gains compiled with the code may not work exactly correct so you may have to try some other values. To input other swing up gains just type the new values when you start up the program. (i.e. BALMID 150 21 4.5 1.165).If the compiled gains do not work you probably only have to adjust the second "Kd" gain. Adjust it in the proximity of the given value (balmid,Kd=21;baltop,Kd=8.8) until link two swings slowly into its balancing equilibrium position. For the mid position you can also try changing the amplitude of the trajectory.

3.2 Writing Your Own Controllers

We have tried to make writing your own DOS controller programs for the Pendubot as easy as possible. There are two ways that we suggest you get started in writing your own controllers. The first way is to start out with our starter source file named "START.C". This program is located in the directory C:\MSI\MSIDEV\PROJ. The windows GUI program uses that file when you run the "Generate New Project" command. That command simply creates a new directory with the project name you specify and copies START.C to that directory renamed as <project name>.C. You can of course do this yourself if you like. Once you have copied and renamed the START.C file into your new project directory you are ready to start developing your new controller.

(Note: in C:\MSI\MSIDEV\PROJ you will also find the file "MSMAKE.BAT" that you will also want to copy. MSMAKE.BAT is a batch file to compile and link your source code.) START.C is filled with

comments to hopefully get you started. You will notice that no PC card library files are linked with the object file during link time. Instead all the drivers for the PC cards are written in the given include files. So if you need to change the functions that interface with the PC cards you can just change the appropriate include file. Again the include files are documented very well so read through them before developing your own controllers.

(Note: the only problem with not creating a library for the PC card driver functions is that if some of the include files call functions from other include files you run into ordering problems.For that reasons you will see a note in the file "DEVPROJ.H" indicating the order in which the card driver files must be included using the command #INCLUDE.)

The second way to start off your new controller is to copy one of the given controller source files and just edit the parts you would like to change. It is much easier to learn by example, so using our source code and modifying only the parts that you want to change can be a very fast way to implement new control algorithms. Always remember that if you are going to run this new controller in a windows DOS Box that you will have to create a PIF file for that controller executable before running it. See *Running Controllers in Windows* section below.

3.3 Compiling Your Own Code

To compile, you first need to setup your DOS environment variable for Microsoft C++ compiling by running the given file "SETMSI.BAT." You will need to check that given batch file to see if the paths indicated in that file are correct for you machine.The contents of that batch file are as follows:

```
@echo on
set TOOLROOTDIR=C:\MSVC
set INCLUDE=C:\MSI\INCLUDE;C:\MSVC\INCLUDE;C:\MSVC\MFC\INCLUDE;%INCLUDE%
set LIB=C:\MSVC\LIB;C:\MSVC\MFCLIB;%LIB%
set INIT=C:\MSVC;%INIT%
set TMP=C:\MSVC\TMP
```

So if you have Microsoft Visual C++ installed in the C:\MSVC directory and you have created the directory C:\MSVC\TMP (you will need to create this directory) and if you installed our software in the C:\MSI directory then this batch file will work to setup your DOS environment for compiling. Otherwise you will have to change the paths that are not correct.

Then to compile you simply type "msmake <your filename>".The batch file msmake will call "cl" and "link" to compile and link your executable. Note:The file MSMAKE.BAT should be copied into your new project directory. You can read MSMAKE.BAT to see what compile flags are being used to create the executable.

Note: The Pendubot's controllers are compiled with the in-line math co-processor option. To recompile the given source code you must make sure that you have installed the in-line math co-processor library, llibc7.lib. In some versions of Microsoft Visual C++ this is not installed with the default settings. During installation choose "Custom" installation, then under the "Libraries" option make sure that all of the following items are checked:

- Memory Module → Large/Huge
- Targets → MS-DOS.EXE
- Math Support → 80×87

After installation, go to the C: `\MSVC\lib` directory and make sure that the file "LLIBC7.LIB" is in that directory.

3.4 Running Controllers in Windows

Running real time controllers in the Windows 3.1 and Windows 95/98 environments is possible but there are limitations and differences between these two operating systems. First of all with both operating systems you must keep in mind that Windows is running underneath your controller and it may suspend your controller. This of course causes your controller's sample rate to be incorrect and in turn, flag an error and cause the control program to exit. Memory to Disk swapping is one thing that happens every once in a while that kills the controllers sample rate. We have found a few guidelines for each operating system that keep these problems to a minimum. We also discuss problems with running in Windows in the "timerbd.h" include file. Please read through those comments also.

Running Controllers in Windows 3.1

In Windows 3.1 you will need to create a PIF file for all your controllers. Use the PIF file editor to create a pif file that

1. indicates the name of your executable,
2. runs in a full screen DOS box (Display Usage: Full Screen),
3. runs in exclusive mode (Execution:Exclusive) and
4. sets as the Start-Up Directory the directory where the executable is located.

See the given controller pif files for examples. Make sure to save the pif file in the same directory the controller is located. You can also make the same changes to the DOSPRMPT.PIF file installed with Windows. This way you can start a DOS session and run your controllers in that DOS session. If you run your controllers with the supplied "MSIDEV" Windows executable you will need a PIF for each controller. This is true for both Windows 3.1 and 95.

Running Controllers in Windows 95/98

The PIF file is different in Windows 95 and unfortunately does not have the exclusive execution property. So for this reason, you will not be able to run certain applications while your controller is controlling the Pendubot. For the most part this is not a problem because there shouldn't be many programs that need to be running while the control is. Matlab is the only one that causes problems.

When you become familiar with my Windows application "MSDEV" you will see that Matlab is used as a graphics engine to plot the Pendubot's response data after a control is finished. (If you do not have a copy of Matlab you will not be able to do this). In Windows 3.1 with the exclusive property active, Matlab is able to run in the background while the controller is running. In Windows 95 this is not the case. The solution is to kill Matlab every time you are done plotting and start it up again when you want to plot your next set of data. Fortunately, if you are running Windows 95 it is more than likely on a 100Mhz machine or higher and it does not take Matlab very long to load.

You will need to create a Win95 PIF file for you controller's application. To do that find your executable as an icon clicking on the appropriate directories in Windows 95 environment. When you find your executable put your mouse on the icon and click the right mouse button and select properties. In the properties dialog box you will need to

1. go to the "screen" tab and check that you want to run in full screen mode.
2. Go to the program tab and enter in the "Working" directory the directory where your executable is located (This way the M-files that your controller creates will be saved in that same directory) and
3. check that you want the program to close on exit.
4. In the Misc. tab, uncheck the item "Allow Screen Saver" if it is not already unchecked. **Note!!! It is a good idea with Windows 95 to disable your screen savers. This way a screen saver will definitely not interrupt your DOS controller.**

3.5 Overview of the 32bit Software Provided

The 32 bit controller software, MSIWIN32.EXE, that we have provided, runs as a 32 bit console application in Windows 95, Windows 98, and Windows NT 4.0. For real-time control applications, the "Real-Time" Priority setting must be used so that other applications will run slowly (or not at all) while the controller is running. Note that problems obtaining reliable sample rates may still occur in some cases. For best results, close all open applications and disable the screen saver before starting MSIWIN32.EXE.

We have supplied you with eight controllers. Six of them are very similar, differing in only the user input. These controllers swing up and balance the Pendubot in the "mid" and "top" operating points. The last two of the controllers implement gain scheduling, controlling the Pendubot along sinusoidal trajectories. We have included all the source code for these controllers, so once you become familiar with the Pendubot you can design your own control algorithms for the system.

The documentation for the controllers is found in each controller's source code. We feel this is a better way for you to learn what our controllers are doing instead of trying to reproduce the code here in this manual, so before you run a controller you will need to study the source code provided. The eight controller executables can be run, like any other console program, by typing the executables name at the command prompt and if desired also type parameter inputs for the controller.

For example to run the BALMID.EXE executable, type at the command prompt: "BALMID DATA.M

150 21 4.5 1.65". BALMID will then run with the given input parameters. If you do not type parameter values or type the wrong number of parameters the default set of parameters are used.

The Graphical User Interface

We have also supplied you with a Windows GUI (or Graphical User Interface), called MSDEV32.EXE, for running the controllers. When you are first developing your own controller you probably will want to run the controllers from a DOS Box. But in a classroom setting or when you are running a number of controllers for demos or experimentation you may find this Windows interface handy. The program, MSDEV32.EXE, allows you to save control files (.CTL) that keep track of the gains or other parameters sent to the controller executable and also interface to the appropriate version of Matlab to plot your response data. More details can be found in the help file entitled README.WIN in the MSIDEV32 directory. We have supplied you with the entire source code for the GUI programs.

The Controller Source Code

The next thing to do is get familiar with the controller's source code (BALMID.C, BALTOP.C, etc.). Study the code so you understand what the controller is doing. Once you feel confident you understand the system you can try the controller programs we supplied. The swing up gains compiled with the code may not work exactly correct so you may have to try some other values. To input other swing up gains just type the new values when you start up the program. (i.e. BALMID 150 21 4.5 1.165).If the compiled gains do not work you probably only have to adjust the second "Kd" gain. Adjust it in the proximity of the given value (balmid,Kd=21;baltop,Kd=8.8) until link two swings slowly into its balancing equilibrium position. For the mid position you can also try changing the amplitude of the trajectory.

3.6 Writing Your Own Controllers

We have tried to make writing your own controller programs for the Pendubot as easy as possible. The first thing you may notice is that the drivers for the PC cards are written in the given include files instead of in a library file, so that if you need to change the functions that interface with the PC cards you can just change the appropriate include file. These include files are well documented so you will find it useful to read them carefully before developing your own controllers.

It is important to note that the order in which the card driver files are #included is important as some of the include files may call functions from other include files. See the note in the file "devproj.h" for further information.

To write your own controller program we recommend that you copy the source code of one of the given controllers and simply change the portions needed to implement your own algorithm.

3.7 Compiling Your Own Code

We recommend using the Microsoft Visual C++ integrated development environment to build your controller applications. The following steps should be used to set up a new project for your controller.

1. Create a new project of the type "Win32 Console Application" as an "empty project".
2. Add your controller's source file to the new project. At the "Project" menu, select "Add to Project", then "Files...".
3. Make sure to switch the "Active Configuration" to "Release". The Outp() and Inp() instructions do not compile in the "Debug" configuration. At the "Build" menu select "Set Active Configuration" and select "Win32 Release".
4. Add an additional include file path so that Visual C++ can find the include files for the drivers. At the "Projects" menu select "Settings". In the settings dialog box click on the "C/C++" tab. Then select the category "Preprocessor". Type the include path

   "C:\MSIWIN32\INCLUDE"

   in the text box labeled "Additional Include Directories". Then select "OK".
5. That is all there is to it. Now you can compile and link your controller's executable code by selecting "Rebuild All" at the Build menu.

Chapter 4

Pendubot Dynamics and Control

In this chapter we discuss the dynamics and control of the Pendubot. The nonlinear equations of motion are described first. This provides a mathematical model that can be used to derive various controllers and to simulate the response. We also provide a parameterization of the Pendubot that can be used to obtain the inertia parameters, and we outline one identification method for estimating these parameters. We also discuss a nonlinear swingup control algorithm based on the idea of partial feedback linearization and a balancing controller using Linear Quadratic Optimal Control theory. These controllers are good starting points to investigate your own algorithms.

4.1 The Dynamics

Since the Pendubot is a two link robot (with only one actuator), its dynamic equations can easily be derived using the so-called Euler-Lagrange equations, which can be found in numerous robotics textbooks, for example [12]. Referring to Figure 4.1 below, the equations of motion are

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = \tau \tag{4.1}$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = 0 \tag{4.2}$$

where q_1, q_2 are the joint angles and

$$\begin{aligned}
d_{11} &= m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2 + 2\ell_1\ell_{c2}\cos(q_2)) + I_1 + I_2 \\
d_{22} &= m_2\ell_{c2}^2 + I_2 \\
d_{12} &= d_{21} = m_2(\ell_{c2}^2 + \ell_1\ell_{c2}\cos(q_2)) + I_2 \\
h_1 &= -m_2\ell_1\ell_{c2}\sin(q_2)\dot{q}_2^2 - 2m_2\ell_1\ell_{c2}\sin(q_2)\dot{q}_2\dot{q}_1 \\
h_2 &= m_2\ell_1\ell_{c2}\sin(q_2)\dot{q}_1^2 \\
\phi_1 &= (m_1\ell_{c1} + m_2\ell_1)g\cos(q_1) + m_2\ell_{c2}g\cos(q_1 + q_2) \\
\phi_2 &= m_2\ell_{c2}g\cos(q_1 + q_2)
\end{aligned}$$

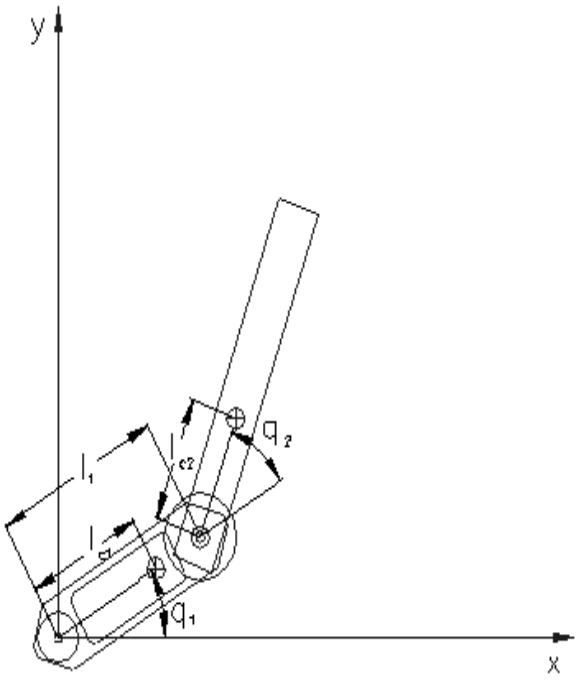


Figure 4.1: The Pendubot

The important distinction then between the system (4.1)–(4.2) and a standard two–link robot is, of course, the absence of a control input torque to the second equation (4.2). For later reference we group the inertia parameters appearing in (4.1)–(4.2) as

$$\begin{aligned} \Theta_1 &= m_1\ell_{c1}^2 + m_2\ell_1^2 + I_1 & \Theta_2 &= m_2\ell_{c2}^2 + I_2 \\ \Theta_3 &= m_2\ell_1\ell_{c2} & \Theta_4 &= m_1\ell_{c1} + m_2\ell_1 \\ \Theta_5 &= m_2\ell_{c2} \end{aligned} \tag{4.3}$$

4.2 The Equilibrium Manifold

Underactuated mechanical systems generally have equilibria which depend on both their kinematic and dynamic parameters; see for example the Acrobot [3]. If the Pendubot is mounted so that the joint axes are perpendicular to gravity, then there will be a continuum of equilibrium configurations, as shown in Figure 4.2, each corresponding to a constant value, $\bar{\tau}$, of the input torque τ. Examining the equations (4.1)–(4.2) we see that the equilibrium configurations are determined by

$$\Theta_4 g\cos(q_1) + \Theta_5 g\cos(q_1+q_2) = \bar{\tau} \tag{4.4}$$

$$\Theta_5 g\cos(q_1+q_2) = 0 \tag{4.5}$$

It is easily seen that, applying a constant torque $\bar{\tau}$, the Pendubot will balance at a configuration (q_1, q_2) such that

$$q_1 = \cos^{-1}(\frac{\bar{\tau}}{\Theta_4 g}) \tag{4.6}$$

$$q_2 = n\frac{\pi}{2} - q_1 \ ; \ \ n = 1, 3, 5, \ldots \tag{4.7}$$

provided $\frac{\bar{\tau}}{\Theta_4 g} \leq 1$.

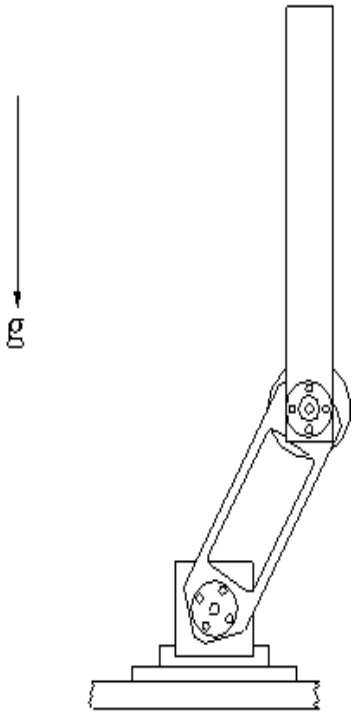


Figure 4.2: Typical Equilibrium Configuration of The Pendubot

4.3 Identification

In addition to the off–line identification of the link inertia parameters using AutoCAD, we performed on–line identification experiments using the Hamiltonian based approach of Gautier and Khalil [5]. The first step in this approach is to write the total energy as the sum of the kinetic energy and potential energy as

$$E = \frac{1}{2}\dot{q}^T D(q)\dot{q} + V(q) \tag{4.8}$$

where $D(q)$ is the inertia matrix defined in (4.1)–(4.2) and $V(q)$ represents the potential energy due to gravity. The key observation of Gautier and Khalil [5] is that the total energy E is linear in the inertia parameters and so may be written as

$$E = W(q, \dot{q})\Theta \tag{4.9}$$

where Θ is the parametrization defined in (4.3). Using the well-known passivity or skew-symmetry property [12]

$$\dot{E} = \dot{q}^T \tau \tag{4.10}$$

we have that, between any two times $t = T$ and $t = T + dT$

$$\begin{aligned} dE &= E(T+dT) - E(T) = \int_T^{T+dT} \dot{q}^T(u)\tau(u)du \\ &= \{W(T+dT) - W(T)\}\Theta \end{aligned} \tag{4.11}$$

Equation 4.11 can be used with a standard least squares algorithm to identify the parameter vector Θ by applying an open loop signal $t \to \tau(t)$ and recording $\tau, q, \dot{q}$ over a given time interval. (See [5] for details). Carrying out this identification procedure using a step input to excite the Pendubot resulted in the following parameter values,

$$\begin{aligned} \Theta_1 &= 0.0308 Vs^2 \\ \Theta_2 &= 0.0106 Vs^2 \\ \Theta_3 &= 0.0095 Vs^2 \\ \Theta_4 &= 0.2087 Vs^2/m \\ \Theta_5 &= 0.0630 Vs^2/m. \end{aligned}$$

The units in the above parameters arise because we first took into account a fixed amplifier gain of $K = 1.2A/V$ and a motor torque constant of $0.4006 Nm/A$. We see that the AutoCAD solid model produced slightly higher numbers which may be due to some over sizing when making assumptions in the model of the Pendubot or static friction which we neglected. Including friction in the identification algorithm produced inconclusive results. The friction in the Pendubot system is very low which may be the reason the energy based identification algorithm does not identify it well. Also, as pointed out in Prüfer, Schmidt and Wahl [7] the energy based algorithm generally has difficulty identifying friction terms because the total energy (the Hamiltonian) is no longer conserved in the presence of friction. The parameters found ignoring friction, as we will demonstrate, work very well in controlling the system. For this reason we did not pursue friction identification further.

4.4 Controlling The Pendubot

In this section we discuss the control of the Pendubot. We first discuss the problem of balancing the Pendubot about one of its equilibrium configurations.

4.4.1 Controllability and Balancing

The balancing problem for the Pendubot may be solved by linearizing the equations of motion about an operating point using a Taylor series expansion of the nonlinear dynamic equations and designing a linear state feedback controller. One very interesting feature of the Pendubot is the continuum of balancing positions. This feature is pedagogically useful in several ways, to illustrate how the Taylor series linearization is operating point dependent and for teaching controller switching and gain scheduling. One

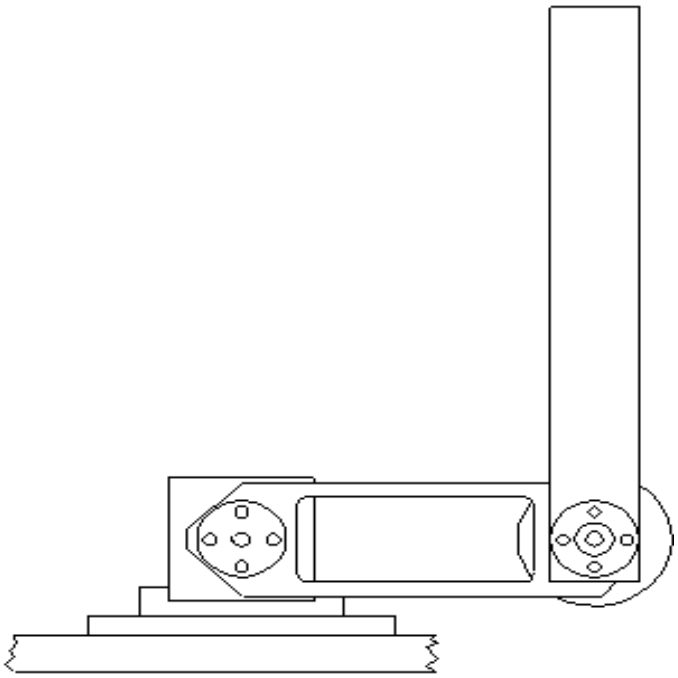


Figure 4.3: Uncontrollable Configurations

can also easily understand physically how the linearized system becomes uncontrollable at $q_1 = 0, \pm\pi$ as illustrated in Figure 4.3. (Note that the zero or reference position for q_1 is horizontal.) As the Pendubot approaches this uncontrollable configuration, the controllability matrix of the linearized approximation becomes increasingly ill–conditioned.

Appendix C contains Matlab code to automatically linearize the nonlinear dynamic equations about an operating point and to generate state feedback gains using LQR and pole placement techniques. This code is included on the disk supplied with the Pendubot.

Figure 4.4 shows the Pendubot balancing at its so-called 'mid' position. This equilibrium is recommended as a first exercise in balance control for students as it is easiest configuration at which to get the Pendubot to balance.

4.4.2 Swing Up Control

The problem of swinging the Pendubot up from the downward configuration to the inverted configuration is an interesting and challenging nonlinear control problem. With this problem one may illustrate the nonlinear control ideas of nonlinear relative degree, partial feedback linearization and zero dynamics. We will give the details of a swing up controller to illustrate this.

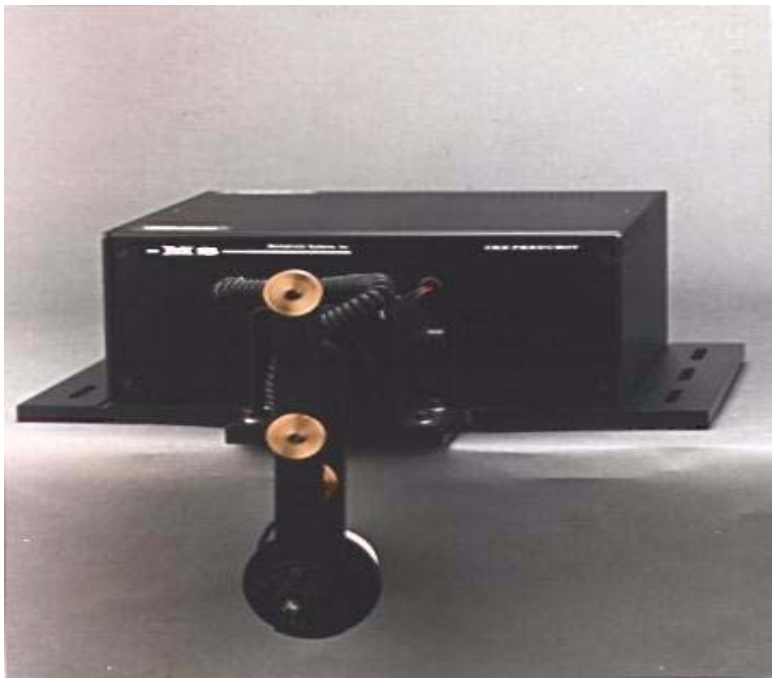


Figure 4.4: The Pendubot balancing at its mid position

4.4.3 Partial Feedback Linearization

We now show how the method of partial feedback linearization can be used to swing up the Pendubot, i.e., to move from the stable equilibrium $q_1 = -\pi/2$, $q_2 = 0$, to the inverted position $q_1 = \pi/2$, $q_2 = 0$. The same approach can easily be modified to swing the system to any point on its equilibrium manifold. For a general treatment of the notion of partial feedback linearization and zero dynamics the reader is referred to the books [6] and [7].

Consider the equation (4.2)

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = 0 \tag{4.12}$$

If we solve for $\ddot{q}_2$ in this equation as

$$\ddot{q}_2 = -\frac{1}{d_{22}}(d_{21}\ddot{q}_1 + h_2 + \phi_2) \tag{4.13}$$

and substitute the resulting expression (4.13) into (4.1) we get

$$\bar{d}_{11}\ddot{q}_1 + \bar{h}_1 + \bar{\phi}_1 = \tau \tag{4.14}$$

where the terms $\bar{d}_{11}$, $\bar{h}_1$, $\bar{\phi}_1$ are given by

$$\begin{aligned} \bar{d}_{11} &= d_{11} - d_{12}d_{21}/d_{22} \\ \bar{h}_1 &= h_1 - d_{12}h_2/d_{22} \\ \bar{\phi}_1 &= \phi_1 - d_{12}\phi_2/d_{22} \end{aligned}$$

A nonlinear feedback linearizing controller can now be defined for equation (4.14) according to

$$\tau = \bar{d}_{11} v_1 + \bar{h}_1 + \bar{\phi}_1 \tag{4.15}$$

The complete system, to this point, is given by

$$\ddot{q}_1 = v_1 \tag{4.16}$$

$$d_{22}\ddot{q}_2 + h_2 + \phi_2 = -d_{12} v_1 \tag{4.17}$$

The additional control term v_1 may now be chosen as

$$v_1 = k_p(q_1^d - q_1) - k_d \dot{q}_1 \tag{4.18}$$

where k_p and k_d are positive gains, so that q_1 tracks the reference angle q_1^d. With state variables

$$\begin{array}{ll} z_1 = q_1 - q_1^d & z_2 = \dot{q}_1 \\ \eta_1 = q_2 & \eta_2 = \dot{q}_2 \end{array} \tag{4.19}$$

the closed loop system may be written as

$$\dot{z}_1 = z_2 \tag{4.20}$$

$$\dot{z}_2 = -k_p z_1 - k_d z_2 \tag{4.21}$$

$$\dot{\eta}_1 = \eta_2 \tag{4.22}$$

$$\dot{\eta}_2 = -\frac{1}{d_{22}}(h_2 + \phi_2) - \frac{d_{12}}{d_{22}} v_1 \tag{4.23}$$

We see from the above that the surface $z = 0$ in state space defines an invariant manifold for the system. Since A is Hurwitz for positive values of k_p and k_d this invariant manifold is globally attractive. The dynamics on this manifold are given by

$$\dot{\eta} = w(0, \eta) \tag{4.24}$$

with

$$w(0,\eta) = \begin{pmatrix} \eta_2 \\ -\frac{1}{d_{12}(\eta_1)}(h_1(0,\eta_1,\eta_2) + \phi_1(q_1^d, \eta_1)) \end{pmatrix} \tag{4.25}$$

It is interesting to note that the same result can be obtained by simply choosing an output equation

$$y = q_1 \tag{4.26}$$

for the original system (4.1)-(4.2), differentiating the output y until the input appears, and then choosing the control input to linearize the resulting equation. The system therefore has relative degree 2 with respect to the output q_1.

The above reduced order system (4.24) therefore represents the zero dynamics of the system with respect to the output $y = q_1$. These zero dynamics are computed by specifying that the y identically track the reference q_1^d. Therefore, substituting $z_1 = 0 = z_2$ into the equations yields

$$\dot{\eta}_1 = \eta_2 \tag{4.27}$$

$$\dot{\eta}_2 = -\frac{1}{d_{22}(q_2)} \phi_2(q_1^d, q_2)) \tag{4.28}$$

Writing the above system in the original second order form yields the expression for the zero dynamics as a second order, autonomous nonlinear system

$$d_{22}(q_2)\ddot{q}_2 + \phi_2 = 0 \tag{4.29}$$

4.4.4 Analysis of the Zero Dynamics

Since we are interested in swinging up the pendulum to the inverted position $q_1 = \pi/2$, $q_2 = 0$, we substitute $q_1^d = \pi/2$ into the equation (4.29) which then can be written using the original description of the system (4.2) as

$$(m_2\ell_{c2}^2 + I_2)\ddot{q}_2 + m_2\ell_{c2}g\sin(q_2) = 0 \tag{4.30}$$

We see that the zero dynamics are just the dynamics of the undamped simple pendulum, which is intuitively appealing, since the control is designed to hold the first link motionless at $q_1 = \pi/2$.

The system (4.30), considered as a dynamical system on the cylinder, has two equilibrium points $p_1 = (0,0)^T$, which is a saddle, and $p_2 = (\pi,0)^T$, which is a center. A typical phase portrait of the system (4.30) is shown in Figure (4.5). Figure (4.5) shows the well-known phase portrait of the simple pendulum.

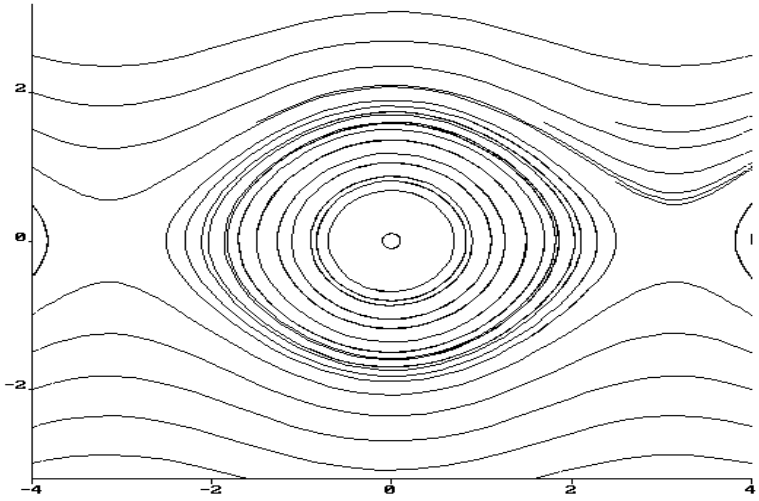


Figure 4.5: Phase Portrait of the Zero Dynamics

It follows that, for initial conditions, $z(0) = z_0$, $\eta(0) = \eta_0$, the state $z(t)$ converges exponentially to zero, while the state $\eta(t)$ converges to a trajectory of the system (4.30). Since almost all trajectories of the system (4.30) are periodic, the typical steady state behavior is for the first link to converge exponentially to $q_1 = \pi/2$ and the second link to oscillate about the equilibrium $(-\pi, 0)$. The strategy for the swing up control is then to excite the zero dynamics sufficiently by the motion of link 1 that the pendulum swings close to its unstable equilibrium and then to switch from the above partial feedback linearization controller to a linear, quadratic regulator designed to balance the pendulum about the vertical.

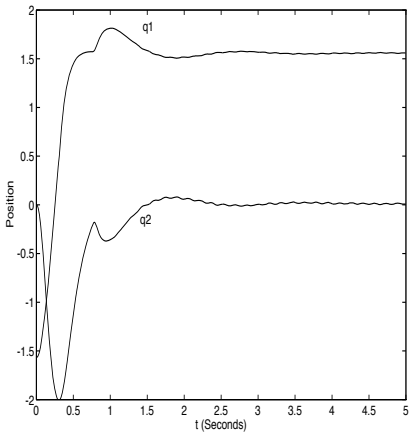


Figure 4.6: Swing-Up and Balance Control at the Top Position.

Figure 4.7: Swing-Up and Balance Control at the Mid Position

4.5 Experiments

This section shows actual responses of the Pendubot system. The balancing controller gains for these runs were found by pole placement methods. Figure 4.6 shows the Pendubot swinging up and balancing at its highest position. Figure 4.7 shows the Pendubot swinging up and balancing at its mid position.

Bibliography

[1] Alleyne, A., et.al., "A Collegewide Laboratory–Based Program in Control Systems Technology at The University of Illinois at Urbana–Champaign," *1996 Conference on Decision and Control*, Kobe, Japan, Dec. 1996.

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

[3] Bortoff, S., and Spong, M.W.,"Pseudolinearization of the Acrobot Using Spline Functions," *IEEE Conf. on Decision and Control*, Tucson, AZ, pp. 593-598, Dec. 1992.

[4] Elmquist, H., *Simnon-User's Guide*, Dept. of Automatic Control, Lund Inst. of Tech., CODEN:LUTFD2/(TFRT-3091), 1975.

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

[6] Isidori, A., *Nonlinear Control Systems*, 2nd Edition, Springer–Verlag, Berlin, 1989.

[7] Nijimeier, and Van der Schaft, *Nonlinear Control Systems*, Springer-Verlag.

[8] Prufer, 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.

[9] Spong, M.W., "Swing Up Control of the Acrobot," *IEEE Control Systems Magazine* Vol. 15, No. 1, pp. 49–55, Feb. 1995.

[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* , pp. 555-556, New Orleans, Dec., 1995.

[11] Spong, M.W., "Energy Based Control of a Class of Underactuated Mechanical Systems," *1996 IFAC World Congress* , San Francisco, CA, July, 1996.

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

[13] Wiklund, M., Kristenson, A., and Astrom, K.J., "A New Strategy for Swinging up an Inverted Pendulum," *Proc.IFAC Symposium*, Sydney, Australia, 1993.

List of Vendors

-Encoders and Encoder Interface Card:

Dynamics Research Corporation
60 Concord Street
Wilmington, Massachusetts 01887-2193
Phone: (508) 658-6100
Fax: (508) 658-0522
Encoder is Model #: F21AC6EDB03-1250
Encoder card is Part #: PC-2B-C-2.00-B-Z698

-Motor:

Minnesota Electric Technology (MET)
1507 First Ave., P.O. Box 3445, Mankato, MN 56001
Phone: 800-373-3166 or 507-625-6117
FAX: 507-625-1485
Part #: 3B-9013182K

-Digital to Analog Output Card and Timer Board:

ComputerBoards, Inc
125 High Street
Mansfield,MA 02048
Phone: (508) 261-1123
Fax: (508) 261-1094
Part #: CIO-DAC02
Part #: CIO-CTR05

-Amplifier and power supply:

Advanced Motion Control
3629 Vista Mercado
Camarillo, California 93012
Tel: (805) 389-1935
Fax: (805) 389-1165
Miniature amplifier for brush servo motor is Part# 25A8
Power supply is Part# PS2X300W-72V