Revitalizing the Furuta pendulum

Hamers, M.R.

Published: 01/01/2004

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. Dec. 2018
Revitalizing the Furuta pendulum

M.R. Hamers

September 2004

Report number: 2004.99

TU/e internship report

Supervisor:

dr.ir. M.J.G. van de Molengraft (TU/e)

University of Technology Eindhoven (TU/e)
Department of Mechanical Engineering
Division Dynamical Systems Design
## Contents

1 Introduction

2 The Furuta pendulum setup
   2.1 Introduction ........................................ 2
   2.2 Sensors and acquisition equipment .................... 2
      2.2.1 Encoder for the actuated rotation .............. 3
      2.2.2 Interpolator for the actuated rotation .......... 3
      2.2.3 Encoder and interpolator of the free rotation ... 3
   2.3 Interface and the TUCDACS .......................... 4
   2.4 Actuation components ................................ 4
      2.4.1 Current amplifier ................................ 4
      2.4.2 AC Motor ...................................... 4
      2.4.3 Transmission .................................... 4
   2.5 Software ............................................. 5
   2.6 Coupling robot and computer ......................... 5

3 The Furuta pendulum model
   3.1 Introduction ......................................... 6
   3.2 Derivation of the dynamical model ................... 7
   3.3 Implementation and use of the model ................ 7

4 Identification of the model parameters
   4.1 Introduction .......................................... 8
   4.2 Frequency Response Function of the pendulum arm .... 8
   4.3 Sliding Mode Control ................................ 9
   4.4 Identification results ................................ 12
   4.5 Directly measured parameters ........................ 13
   4.6 Verification ......................................... 14
   4.7 Friction Compensation ................................. 15

5 Controlling the Furuta pendulum
   5.1 Introduction ........................................... 16
   5.2 Deriving the controllers .............................. 16
      5.2.1 Swing-up by energy control ...................... 16
      5.2.2 Linearized controller ............................ 18
      5.2.3 Input/Output linearizing controllers ............ 20
   5.3 Simulation and experimental results .................. 24
      5.3.1 Swing-up controller + linearized controller ..... 25
      5.3.2 I/O linearizing controller β + linearized controller 26
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6 Controller switching</td>
<td>29</td>
</tr>
<tr>
<td>6.1 Introduction</td>
<td>29</td>
</tr>
<tr>
<td>6.2 Switching between modes</td>
<td>30</td>
</tr>
<tr>
<td>6.3 Switching between stages</td>
<td>31</td>
</tr>
<tr>
<td>7 Manual for the Furuta pendulum</td>
<td>32</td>
</tr>
<tr>
<td>7.1 Files and folders</td>
<td>32</td>
</tr>
<tr>
<td>7.2 Starting up the demonstration experiment</td>
<td>33</td>
</tr>
<tr>
<td>7.3 Pendulum data</td>
<td>34</td>
</tr>
<tr>
<td>8 Conclusions and recommendations</td>
<td>35</td>
</tr>
<tr>
<td>Bibliography</td>
<td>36</td>
</tr>
<tr>
<td>A Datasheets</td>
<td>37</td>
</tr>
<tr>
<td>A.1 Heidenhain ERN 480 rotational encoder</td>
<td>37</td>
</tr>
<tr>
<td>A.2 The Dynasyn DV7-6-4M AC motor</td>
<td>37</td>
</tr>
<tr>
<td>A.3 Leonard+Bauer GEL 214 interpolator</td>
<td>38</td>
</tr>
<tr>
<td>B Derivation of the dynamical model</td>
<td>42</td>
</tr>
<tr>
<td>C The Sliding Mode Controller</td>
<td>45</td>
</tr>
<tr>
<td>C.1 Derivation of control law</td>
<td>46</td>
</tr>
<tr>
<td>C.2 Derivation of adaptation mechanism</td>
<td>47</td>
</tr>
<tr>
<td>C.3 SMC results</td>
<td>50</td>
</tr>
<tr>
<td>D Simulink models</td>
<td>51</td>
</tr>
<tr>
<td>D.1 Coupling of pendulum to computer</td>
<td>51</td>
</tr>
<tr>
<td>D.2 Total pendulum model</td>
<td>52</td>
</tr>
<tr>
<td>E Linearisation of the non-linear model</td>
<td>53</td>
</tr>
<tr>
<td>E.1 Linearisation for upright position</td>
<td>53</td>
</tr>
<tr>
<td>E.2 Linearisation for downward position</td>
<td>54</td>
</tr>
<tr>
<td>F Input/Output Linearisation</td>
<td>56</td>
</tr>
<tr>
<td>F.1 Design of controller for $\alpha$</td>
<td>56</td>
</tr>
<tr>
<td>F.2 Design of controller for $\beta$</td>
<td>59</td>
</tr>
<tr>
<td>G Simulation and experimental results</td>
<td>62</td>
</tr>
<tr>
<td>G.1 Energy controller + I/O linearizing controller $\alpha$</td>
<td>62</td>
</tr>
<tr>
<td>G.2 I/O linearizing controller $\beta$</td>
<td>64</td>
</tr>
<tr>
<td>G.3 Linearised controller $\beta = 0$</td>
<td>69</td>
</tr>
<tr>
<td>H Switching mechanism code</td>
<td>71</td>
</tr>
<tr>
<td>H.1 Switching_mechanism.c</td>
<td>71</td>
</tr>
<tr>
<td>H.2 Sequence.h</td>
<td>78</td>
</tr>
<tr>
<td>I Accompanying CD-ROM</td>
<td>79</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The rotary inverted pendulum or Furuta pendulum is a typical example of an underactuated system. It means that the robot has more degrees of freedom than actuators, assuming the various pendulum components to be rigid. Such systems are very popular for educational purposes, because control of underactuated systems mostly requires some extra tools or tricks than used in 'normal' control problems.

The goal of this internship is to 'revitalize' the Furuta pendulum, located in the D&C laboratory at the TU/e, and to write a new manual for it. Revitalization is needed because the current setup is old-fashioned and has to be modernized. A reference manual for this new setup enables first time users to learn how to operate and control the pendulum. This report will serve as a manual as well. Moreover a demonstration of the Furuta pendulum, executing different modes, is needed to be able to show the possibilities of the robot. In chapter 2 first the new setup of the robot and all its components are discussed. Before designing a balancing controller a mathematical model has to be designed. In chapter 3 the model describing the dynamics of the Furuta pendulum is derived. This will be done using the Lagrange's equations of motion. The model is quite complex because the robot suffers from severe non-linear behavior due to Coriolis and centrifugal forces and the presence of stick-slip friction. The values of the model parameters determine to what extent the model is able to represent the dynamics of the real robot. In chapter 4 the model parameters that can not be measured directly, are estimated using a adaptive controller. A Coulomb friction compensation is designed in this chapter also. Once the model is complete, controllers can be designed to make the pendulum perform the different modes in the demonstration experiment. In chapter 5, five model-based controllers are designed. The controllers are all individually responsible for one or more parts of the different modes. So control has to be switched between the controllers. The switching strategy and its implementation will be discussed in chapter 6. Finally in chapter 7 a few final comments for first-time users will be made in order to enable them to operate the Furuta pendulum.

Because of the complexity of the pendulum model, some controllers can not be modeled easily using standard Matlab Simulink blocks. These controllers therefore have to be programmed in C-code. The learning process to be able to write programs in C-language will not be discussed in this report.

This internship is performed at the TU/e within the Dynamics and Control group under the supervision of dr.ir. M.J.G. van de Molengraft.
Chapter 2

The Furuta pendulum setup

2.1 Introduction

The setup of the Furuta pendulum is schematically represented in figure 2.1. Two encoders, of different type, for each of rotational degree of freedom are connected to a linear interpolator. The interpolators improve the resolution and therefore the accuracy of the sensors. Then the interpolator signals are sent to an interface, which counts the individual pulses of the interpolator and converts the signal to be compatible with the TUEDACS. From the TUEDACS the signal is directly fed into the computer. Only one degree of freedom can be actuated directly by a motor. The motor is controlled successively via the computer, the TUEDACS, the interface and the current amplifier. A planetary transmission is present between the motor axis and the actuated arm of the pendulum. In the next sections every component, its functions and characteristics will be discussed separately. For safety measures the Furuta pendulum is equipped with an emergency button. As long as an experiment runs, this button must be pressed down. If the button is released during an experiment, the pendulum will immediately stop responding.

Figure 2.1: The setup of the Furuta pendulum
CHAPTER 2. THE FURUTA PENDULUM SETUP

2.2 Sensors and acquisition equipment

In this section the encoders as well as all the equipment needed to transform encoder signals into angle information are discussed. If available, reference to more information about the components will be given.

2.2.1 Encoder for the actuated rotation

The encoder used to measure the rotation of the vertical and actuated degree of freedom is an incremental optical encoder of Heidenhain, type ERN 480. An optical encoder has a thin plate of glass with a radial grating of lines. For this type the grating consists of 5000 lines, leading to 5000 line counts per revolution. One grating period is defined as the distance between two succeeding lines. At a small distance, a scanning reticle is placed parallel to this glass plate. It has four grated fields, phase shifted over 90° with respect to each other. Each field on the scanning reticle is enlightened by a beam of collimated light. For each grating field a light sensor measures the amount of light that goes through the grating field and glass plate successively. As a result of the phase shift between the four scanning fields the output of the light sensors are four sine-waves, each shifted over 90°. The encoder will, after processing the four signals, output 2 sine functions with a phase shift of 90°. These signals can be processed further by, for example, an interpolator, to extract rotational information. For more information about rotary encoders see [Hei04]. A datasheet for the encoder used in the setup is presented in Appendix A.

2.2.2 Interpolator for the actuated rotation

A linear interpolator is able to acquire angle information out of two sine-waves generated by an encoder. It easy to understand that when two sine waves are phase shifted by 90° and only the sign of a signal can be measured, four states in each grating period can be distinguished. So the minimum accuracy of the encoder is 1/4 of a grating period. This type of interpolation corresponds to an interpolation factor of 1. The interpolator used in this setup is a Leonard+Bauer type GEL 214 and is able to interpolate up to a factor of 40. During all the experiments it showed sufficient to work with an interpolation factor of 1. The output of the interpolator in this configuration is thus 2 x 10^4 pulses per revolution of the motor axis. For all experiments in this report a value of 1 is used. More information about this type of interpolator is given in Appendix A.

2.2.3 Encoder and interpolator of the free rotation

The vertical, non-actuated degree of freedom is also equipped with an incremental rotary encoder. Unfortunately less documentation about the encoder itself as well as the interpolator is present. It is most likely that the encoder and the interpolator are home-made. This can also be concluded from the fact that the bearings of the pendulum, its suspension and the encoder are integrated. Moreover, type-numbers or brand marks printed on the outside of the encoder or the interpolator are not present. Because the pendulum arm is rotating the measured data can not be transported to the acquisition board, through a normal wire. To overcome this problem a hollow axis is placed through the pendulum arm, the transmission set, the optical encoder and the motor axis. The wires coming from the encoder are connected to brushes on the outside of the hollow axis. Through wires inside the hollow axis the data is further transported. From a simple experiment, turning the pendulum rod over 360°, it can be concluded that the encoder outputs 4000 pulses per revolution.
2.3 Interface and the TUeDACS

The interface box used in the experimental setup is homemade and no documentation is available. The interface is transferring the data of the interpolators, pulses, to a form which the counters of the TUeDACS can handle. In the TUeDACS the pulses are counted and sent to the computer for further processing.

The interface also has a function for the actuation of the pendulum arm. The TUeDACS can output a voltage of maximal ±2.5V. The amplifier needs an input voltage of ±10.0V. The interface transforms the output signal of the TUeDACS to a signal suitable for the current amplifier, in a linear way. For example a voltage of −2.5 V of the TUeDACS corresponds to a voltage of −10.0 V send to the amplifier.

2.4 Actuation components

This section deals with all components responsible for the actuation of the pendulum arm; thus the amplifier, the motor and the planetary transmission.

2.4.1 Current amplifier

The current amplifier in the setup of the pendulum is an AMK Pumasyn type Pus-3. Also for this component no documentation is available. In [Hen02] there is a short description of this amplifier or source-inverter. The amplifier needs an input voltage between 10.0 V and −10.0 V. This input signal, a measure for the desired torque delivered by the motor, is translated into three phase shifted signals with a fundamental frequency. The actual torque delivered by the motor is controlled by this source inverter by means of ‘Pulse Width Modulation’ and makes sure that the delivered torque or current equals the desired value. The amplifier has a linear behavior between input voltage and output motor torque for angular velocities up to 12 rad/s.

2.4.2 AC Motor

The actuator of the pendulum arm is a Dynasyn type DV7-6-4M. The only source of information is a little tag on the AC motor itself. It gives the nominal characteristics of the motor. These are presented in Appendix A. It is very important to know the (linear) transfer of the amplifier-motor combination from input voltage to output torque. In [Hen02] it was found that the amplifier-motor combination had an transfer of 16 Nm/V. The total linear transfer from output voltage of the computer, or the TUeDACS, to motor torque is then 64 Nm/V, because the interface amplifies the output voltage with a factor of 4.

2.4.3 Transmission

The planetary transmission is a Alpha Gear type SPF"M", see [Hen02]. Because the transmission is fully integrated in the pendulum setup, it is not reachable anymore and this information could not be verified. See [Alp04] for more information about these planetary transmissions. The type used in the experimental setup is most likely out of production and no information was available on the website. The transmission ratio is exactly 8.192. With this in mind and the fact that the optical encoder for the actuated rotation gives $2 \times 10^5$ pulses per revolution of the motor axis, it can be concluded that the accuracy of the arm rotation equals $3.84 \times 10^{-5}$ rad.
2.5 Software

All simulations and experiments during this internship have been done using Matlab Simulink in combination with the TUeDACS. Because the model of the rotary inverted pendulum is quite complex it is not possible to model it easily with the standard Simulink blocks. Also the controllers that will be designed, and a sort of switching mechanism to switch between them, are of such complexity that another approach of modeling is desired. So-called S-functions are used to make a model of the pendulum and its controllers. S-functions have been made especially to be used in Simulink. The S-function layout is standard. Each S-function has the same 'modes', like 'initialization' or 'update', in which it can operate. Simulink just calls the function in a certain mode, which can be different every call. S-functions can be written in standard Matlab language or C. In simulations the language used to write the S-function is not of great importance. The C-version will probably be a little bit faster. On the other hand, for the experiments the importance of the used language is very high. For each experiment the Simulink model has to be compiled to a stand-alone application which can communicate with the acquisition equipment. If the standard Matlab language 'm' is used, the building procedure of the model needs, besides your written S-function, a so called TLC-file. This TLC-file contains the information to convert your S-function to a C-program, which your acquisition system can understand. This is a very difficult file to write because you exactly need to know how the acquisition system works. A way to overcome writing such a file is to write your S-function directly in the C-language. Simulink provides templates of those C-file S-functions. Once you understand the structure of a C-file S-function, you can directly program your C-code into such a template file. After compiling your written S-function to a application and using a 'S-function' block in Simulink the model can be build to a real-time stand-alone application needed to run the experiments. It is decided to write all the S-functions for the experiments as well as the simulations directly in C-code.

2.6 Coupling robot and computer

The pendulum is connected with the computer through the TUeDACS. The real pendulum is modeled as a subsystem in Simulink with one input, the motor torque, and four outputs, both rotations of the pendulum and their derivatives. The input signal in the Simulink model is first multiplied with the inverse of the transfer from computer to motor torque. Then it will appear as a torque at the motor axis again. Both the signals from the encoders are multiplied with $2\pi$ and divided with the number of pulses per rotation to get the angle in radians. The derivatives are computed by differentiating the position signals. The encoder signals are discrete signals and therefore the derivatives will contain a lot of noise. They need to be filtered before they can be used. The filtering is done with 4th-order Butterworth filters. The cut-off frequency for the actuated rotation is 50 Hz and the cut-off frequency for the free rotation is 90 Hz. The implementation of the real pendulum and the coupling between the computer and the TUeDACS is given in Appendix D.
Chapter 3

The Furuta pendulum model

3.1 Introduction

The Furuta pendulum is schematically represented in figure 3.1. The robot has two degrees of freedom, a forced rotation around the vertical axis and a free rotation around an axis perpendicular to the driven axis. The system is so-called underactuated, which means that the system has less actuators than degrees of freedom. The rotation of the pendulum arm, $\alpha$, is forced by the actuator and the rotation of the pendulum rod, $\beta$, is free. The length, $l_1$, represents the distance between the vertical axis of rotation and the point where the pendulum rod is mounted on the arm. The length, $l_2$, is defined as the distance between the latter point and the center of mass of the pendulum rod.

![Figure 3.1: Schematic representation of the Furuta pendulum](image)

The arm has a mass $m_1$ and a moment of inertia $J_1$ around the actuated vertical axis. The pendulum rod has mass $m_2$ and a moment of inertia $J_2$ around its center of mass.
CHAPTER 3. THE FURUTA PENDULUM MODEL

3.2 Derivation of the dynamical model

To derive the dynamical model of the Furuta pendulum, the Lagrangian approach is used. First a set of generalized coordinates has to be determined. The system has two degrees of freedom so a possible set of generalized coordinates is very straightforward and is chosen as:

\[ q = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]  

(3.1)

Here \( \alpha \) and \( \beta \) are the positions of the actuated arm and the pendulum rod respectively, as defined in figure 3.1. The downward position of the pendulum rod is considered as \( \beta = 0 \). The derivation of the model itself is presented in Appendix B. Finally, it is possible to write the dynamical behavior of the pendulum in so-called state-space format, which will turn out to be very useful for the implementation in the software.

\[ \begin{align*}
\dot{x} &= \begin{bmatrix} \ddot{q} \\ \dot{q} \end{bmatrix}, \\
\dot{z} &= \begin{bmatrix} \ddot{q} \\ \dot{q} \end{bmatrix} = \left[ \begin{array}{c} M^{-1} (fu - D\dot{q} - F) \end{array} \right]
\end{align*} \]  

(3.2)

with:

\[ M = \begin{bmatrix} J_1 + m_2l_2^2 + m_2l_2^2 \sin^2 \beta & m_2l_1l_2 \cos \beta \\ m_2l_1l_2 \cos \beta & J_2 + m_2l_2^2 \end{bmatrix} \]

\[ D = \begin{bmatrix} b_1 + m_2l_2^2 \beta \sin \beta \cos \beta & m_2l_1l_2 \beta \sin \beta \cos \beta - m_2l_1l_2 \beta \sin \beta \\ -m_2l_1l_2 \beta \sin \beta \cos \beta & b_2 \end{bmatrix} \]

\[ F = \begin{bmatrix} C_1 \text{sign}(\dot{\alpha}) \\ m_2g l_2 \sin \beta \end{bmatrix} \]

\[ f = \begin{bmatrix} K_u \\ 0 \end{bmatrix} \]

With \( q \) as defined in equation 3.1.

3.3 Implementation and use of the model

The equations of motion for the pendulum are now complete and ready for implementation in the software. To implement the model of the pendulum for use in the simulations, a so-called m-file S-function is written. See chapter 2 for more information about these functions. In this S-function, the model was implemented as equation 3.2. Every time the model S-function is called by Simulink all the matrices \( M, D, F \) and \( f \) are evaluated and output is generated. The model is also used to design some of the controllers for the pendulum. This is done in chapter 5.

The model in equation 3.2 and the model finally used in simulations are slightly different. All experiments performed on the real pendulum are done with a (simple) compensation for the Coulomb friction. The model used in simulations should therefore also be free of Coulomb friction. This is because the controllers are very sensitive to model perturbations. When this compensation is neglected, the highly non-linear behavior of the Coulomb friction can restrain the used controller strategy to work. Therefore the model has to match the dynamical behavior of the real pendulum as good as possible. The part of \( F \) with the Coulomb friction term is deleted, forming the final model of the Furuta pendulum used during simulations. For the sake of completeness the dynamical model was first extended with a Coulomb friction term to be deleted again after the compensation was implemented. It is very important to be aware of the fact that if the friction compensation is not implemented or turned off, the model and also all derived controllers will have to be recomputed. In chapter 4 the Coulomb friction compensation is designed.
Chapter 4

Identification of the model parameters

4.1 Introduction

This chapter deals with the estimation or measurement of the model parameters. It is very important to know the parameters of the model, derived in chapter 3, as accurate as possible. The quality of the controllers that will be constructed in chapter 5 entirely depends on the accuracy of the model and the model parameters. To estimate the pendulum arm parameters, only the rotation of the pendulum arm itself is considered. And the pendulum rod is actually dismounted from the robot. First of all a transfer function, from input torque to output angle of the pendulum arm, is measured. This is done in section 4.2. The parameters of the stripped setup are estimated with a so-called SMC\(^1\). A SMC is a somewhat unusual parameter estimation procedure. Because it was already used successfully on a comparable robot, it is also applied to the Furuta pendulum. In section 4.3 the principles of a SMC will be discussed briefly. The results of the parameter estimation will be presented in section 4.4 and because they are satisfactory there is no need to use a more common parameter estimation procedure. The parameters of the pendulum rod, mass and inertia, do not need to be estimated. The mass can be measured and the inertia can be computed. The only requirement to make sure this is legitimate is that all estimated parameters are in SI units as well. This means that the transfer from output voltage to torque has to be known or measured and expressed in SI units. This also means, when the right transfer is used during the identification with the SMC, that all estimated parameters will be SI. In chapter 2 it is concluded that the transfer amounts to \(64 \text{ Nm/V}\). In section 4.6 this will be verified, to make sure the estimated and measured model parameters can be used in the same model. In the last section, 4.7, the simple Coulomb friction compensation model is derived and compared to the measured friction.

4.2 Frequency Response Function of the pendulum arm

As a first exploration of the system, a FRF for the pendulum arm is measured. Because the arm shows obvious presence of Coulomb friction this cannot be done by the use of white noise as only input. Therefore the measurement is executed in closed loop. To overcome the non-linearity of Coulomb friction the system is brought into so-called jogmode. The arm moves with a constant positive or negative velocity. To realize this, the setpoint for the pendulum arm is chosen to have a constant speed, starting in 0 at \(t = 0\). Because the measurement is performed in close loop the simple PD controller assures the pendulum arm to track the setpoint. Then white noise is entered in the loop between the controller and the plant, see figure 4.1. The transfer from this

\(^1\)Sliding Mode Controller
white noise to the plant input is called the sensitivity function. With this sensitivity function the desired transfer function of the plant can be directly derived.

\[ S = \frac{1}{1 + PC} \Rightarrow P = \frac{S^{-1} - 1}{C} \] (4.1)

To make sure the controller does not affect the pendulum arm dynamics and moreover to obtain proper measurement data in the low frequency range, its bandwidth should be as low as possible. It is important though that the P and D actions just have to be strong enough to make sure the pendulum arm tracks the setpoint and the velocity does not change sign. This is needed to avoid the non-linear behavior of the Coulomb friction. After some tuning the values for the proportional and differential action are chosen 2 and 1 respectively, resulting in a bandwidth of 2 Hz. The results of the measurements are given in figure 4.2. The transfer function shows a double integrator behavior till approximately 150 Hz, where the first eigenfrequency is located. The second and third eigenfrequency are located at 230 Hz and 290 Hz respectively. From this measurement a first indication for the inertia of the arm can be estimated. In the low frequency range the FRF can be approximated by \( P(s) = \frac{1}{J_1 s^2} \). For the moment of inertia of the pendulum arm follows \( J_1 \approx 0.3 \, kgm^2 \). By looking at the phase of the FRF, a considerable time-delay, reaching from \(-180^\circ\) at 10 Hertz to \(-720^\circ\) at 350 Hertz, is present. This is due to the dynamics of source inverter. The coherence function of this measured FRF, which is depicted in figure 4.3, shows a good noise-response ratio for frequencies not lower as approximately 1 Hz.

4.3 Sliding Mode Control

In this section the working principles of a SMC will be discussed briefly. A Sliding Mode Controller is actually a special case of a greater class of adaptive controllers, the MRAC\(^2\). The schematic representation of such a controller is presented in figure 4.4. The basis of a MRAC is that the controller parameters are being adapted, by an adaptation mechanism, in order to make the controlled system behave like a user defined reference model. This reference model can be seen as a filter that makes sure the system can follow the generated setpoint. It is usually chosen to be of the same order as the plant and in this case the reference model is a standard mass-spring-damper system characterized by two parameters \( \xi_m \) and \( \omega_m \). The adapting parameters in the controller are usually model parameters to be estimated. The controller generally consists of a full dynamics feedforward term, based on the estimators, and a part based on the tracking error. The adaptation mechanism tries to minimize the control effort of the error based part by adapting the model parameters and improving the feedforward term. So a standard MRAC consist actually of two dynamic components. The first one is the controller itself which makes sure the system tracks the reference signal, and in the ideal case brings the error to zero for \( t \to \infty \). The second part is the

\(^2\)Model Reference Adaptive Controller
CHAPTER 4. IDENTIFICATION OF THE MODEL PARAMETERS

Figure 4.2: Measured frequency response function

Figure 4.3: Coherence of measured FRF
adaptation mechanism which tries to converge the estimators of the model parameters to the real parameters of the plant. The controller in this form though, will only lead to satisfactory results if the model represents the real plant exactly and when the measurements are noise-free. Otherwise the adaptation mechanism may change the parameters based on incorrect data and they will not converge sufficiently or show drift. In a SMC some measures have been taken to overcome this problem. These measures and the derivation of the SMC itself are presented in Appendix C. The SMC finally used to estimate the parameters of the Furuta pendulum is mathematically described in equations 4.2 and 4.3.

\[ \dot{\theta} = -\Gamma^{-1} W^T s \]  
\[ T_m = -\frac{1}{S_2} J_1 e_\alpha + W \dot{\theta} - \Lambda \text{sat}(s, \phi) \]  

where:

\[ W = [ \dot{\alpha}_m \ \dot{\alpha}_m \ \text{sign}(\dot{\alpha}_m) ] \]
\[ \dot{\theta} = [ \dot{J}_1 \ \dot{b}_1 \ \dot{C}_1]^T \]
\[ s = S_1 e_\alpha + S_2 \dot{e}_\alpha \]

with:

- \( \dot{\theta} \): A column with estimated model parameters
- \( \Gamma \): A 3 x 3 regular diagonal matrix which controls the adapting speed
- \( W \): A row with feedforward dynamics
- \( s \): The switching parameter
- \( J_1, b_1, C_1 \): Estimated parameters
- \( S_1, S_2 \): Switch parameters
- \( \Lambda \): Gain representing model errors and noise
- \( \phi \): Saturation level for switching parameter
- \( \alpha_m, \dot{\alpha}_m, \ddot{\alpha}_m \): Reference signal and derivatives
- \( e_\alpha, \dot{e}_\alpha \): Tracking error and derivative
CHAPTER 4. IDENTIFICATION OF THE MODEL PARAMETERS

For the SMC to work properly all the estimators need an initial value. Of course the better the initial value approaches the real value, the faster the parameters will converge. All the parameters $\Gamma, S_1, S_2$ and $\Lambda$ have to be chosen carefully to make sure the system is stable. For example if the initial values of the parameters are not chosen well and $\Gamma$ is chosen too high, the parameter updates may become unstable. Or if $\Lambda$ is chosen too small, the system is not able to stay on the switching manifold and may chatter or become unstable. Also if $S_2$ is chosen too large the switching parameter is highly dependent on the velocity error and therefore on the noise of the position measurement and its filters. This can also be a source of instability. The reference model plays a role in the stability of the total system also, see Appendix C. To get good results it is often wise to start carefully, so a low $\Gamma^{-1}$ and a relative high $\Lambda$. A few experiments can be performed with these parameter values, and the estimators of one experiment can be used as initial values for the next. After each experiment the controller can be set a little bit stiffer to decrease the tracking errors. Under the assumption that the system will stay in 'Sliding Mode', see Appendix C for an explanation, the last term in equation 4.3 is just a PD controller. Therefore the first term, which is also a derivative action can be combined with the derivative action following from the last term with the saturation.

4.4 Identification results

In this section the results of the parameter identification of the pendulum arm with the SMC will be presented. First a standard MRAC is used to determine a first estimate for the pendulum arm parameters. After that the SMC is used to increase the accuracy of the estimators. In Appendix C all results are presented in figures as well as the chosen values for the SMC parameters. First the experiments are executed with a relatively high $\Gamma$ and low $\omega_m$ to allow estimator updates. After a few successive experiments the estimator update reduces, because they are already close to the real value, and the bandwidth of the reference model is increased to reduce the tracking error. The better the estimated parameters are, the less they will change in time and the smaller the tracking error will be. Once again, when the system is in sliding mode, the control law consist of a full dynamics feedforward, based on the estimates, and a PD control term on the tracking error. So a measure for the accuracy of the parameters is the tracking error, given of course that the model of the plant fits the real system well enough. A few model assumptions are made with respect to the real system. In the model used to compose the SMC, the Coulomb friction is assumed to be symmetric and position independent. Also the viscous damping is assumed to be position independent. The real Furuta pendulum shows an obvious presence of position and direction dependency. But because the model has only one parameter to characterize Coulomb or viscous friction, these effects will be leveled out. The parameters used during this experiment are listed in table 4.1. During this final experiment the tracking error is at most 1.5 mrad. After switching into 'sliding mode' the parameter $s$ stays within the saturated band specified by $\phi$. The estimated values for the moment of inertia, viscous friction and Coulomb friction are given in table 4.2 and will be verified in the next section.
CHAPTER 4. IDENTIFICATION OF THE MODEL PARAMETERS

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ξm</td>
<td>0.7</td>
</tr>
<tr>
<td>ωm</td>
<td>40π</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>φ</td>
<td>0.015</td>
</tr>
<tr>
<td>Γ</td>
<td>diag(50)</td>
</tr>
<tr>
<td>S1</td>
<td>40</td>
</tr>
<tr>
<td>S2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: SMC parameters for the final experiment

<table>
<thead>
<tr>
<th>Parameter Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1 [kgm²]</td>
</tr>
<tr>
<td>b1 [Ns/rad]</td>
</tr>
<tr>
<td>C1 [Nm]</td>
</tr>
</tbody>
</table>

Table 4.2: Identification results for the SMC

4.5 Directly measured parameters

The parameters of the pendulum rod, its mass and moment of inertia can be measured directly. The pendulum rod is weighted and its dimensions are measured. With these parameters the moment of inertia of the pendulum rod around its center of mass can be computed. The dimensions of the pendulum arm are measured also. The results of the directly measured and computed parameters are given in table 4.3. The only parameter that can not be measured directly and is not estimated by the SMC is the viscous friction of the pendulum rod. But because this viscous damping is so small it is not of great importance. It is assumed that the viscous friction equals $b_2 = 1.0 \times 10^{-4}$.

<table>
<thead>
<tr>
<th>Parameter Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2 [kg]</td>
</tr>
<tr>
<td>l_oe2 [m]</td>
</tr>
<tr>
<td>l2 [m]</td>
</tr>
<tr>
<td>J2 [kgm²]</td>
</tr>
<tr>
<td>l1 [m]</td>
</tr>
</tbody>
</table>

Table 4.3: Results of directly determined parameters
4.6 Verification

In the Simulink model of the SMC the input to the real pendulum is assumed to be a torque. This means that the desired torque from the model has to be multiplied with the inverse of the constant transfer from computer to motor. Then the signal is outputted by the computer and goes through all components to the motor and finally appears at the pendulum arm as a torque again. To verify this constant transfer, a differential measurement is done. If the transfer is correct, the estimated, directly measured and computed parameters can be used in the same model of the Furuta pendulum. The moment of inertia, estimated by the SMC is changed by a known amount. Then the experiment with the SMC is repeated to estimate the parameters again. If the estimator of the inertia has changed with the computed amount, represented in SI, it can be concluded that the used transfer is expressed in SI. The moment of inertia is increased by placing a circular disc with radius $r = 115$ mm and a mass of 1 kg at distance of 175 mm of the axis of rotation of the pendulum arm. It can be computed that for each disk the moment of inertia is increased with 0.032 kgm². In total two experiments have been done, with one and with two extra disks. The results of these experiments are presented in figure 4.5. From this figure it is obvious that the estimator of the moment of inertia increases with the computed amount. Therefore it can be concluded that the transfer from computer to torque, $K_s = 64 [Nm/V]$ is correct.
CHAPTER 4. IDENTIFICATION OF THE MODEL PARAMETERS

4.7 Friction Compensation

In [Hen02] a few types of friction compensation were already constructed for the pendulum arm. Because of the fact that they were designed using a different acquisition board and neglecting the transmission ratio, the results are not compatible with the new setup. It is decided to design a new, simple, Coulomb friction compensation suitable for the purposes of the pendulum during this internship. From the estimator for the Coulomb and viscous friction a model of the total friction can be derived. This model is compared to the measured torque needed to rotate the arm at a constant speed. A constant velocity setpoint in combination with a weak PD controller is used to perform this experiment. The experiment is executed for 20 different rotating speeds. After averaging the controller output per measurement the results are compared with the friction model. This is represented in figure 4.6. The measurements show a clear existence of so-called Stribeck friction. For low speeds the friction increases in contrast with what can be expected from viscous friction. From the figure it can be concluded that the friction model is consistent with the measurement for absolute speeds of approximately 0.5 m/s and higher. For low speeds the model does not match the measurements very well. It is decided though, to implement a normal Coulomb friction compensation, \( C \cdot \text{sign}(\dot{\alpha}) \), to get rid of the main part of the non-linear behavior.

To overcome numerical problems due to the sign function in the compensation a sigmoid function is used:

\[
M_{fc} = C_1 \left( 1 - \frac{2}{e^{200\alpha} + 1} \right) \tag{4.4}
\]

The pendulum uses its own, actual velocity information to compensate for the Coulomb friction. Of course much better compensation mechanisms can be derived, for example position and speed dependent compensations. After implementing the simple compensation in Simulink and studying the results it was decided that there was no need for improvement.
Chapter 5

Controlling the Furuta pendulum

5.1 Introduction

In this chapter the controllers used to control the pendulum are being derived and evaluated. First in section 5.2 the controllers are constructed. Section 5.3 deals with the simulations and experiments with the controllers. First of all a so-called energy based controller is designed to be able to swing up the pendulum rod in upright position. Energy based controllers have already been designed for other Furuta pendulum setups, see [Arn03] and [Ber03]. Because this swing-up controller is not able to stabilize the pendulum, a linearized controller is computed to balance the pendulum rod. This is also done for the downward position. For each degree of freedom Input/Output linearizing controllers are derived to make the pendulum arm or pendulum rod follow a predefined setpoint. With all these controllers the pendulum rod is able to perform different movements referred to as pendulum modes, which are designed for the demonstration experiment. By executing these pendulum modes successively the final demonstration experiment can be designed. This is discussed in chapter 7. To be able to combine the different pendulum modes into a single demonstration experiment a switching strategy must be designed and programmed. This switching algorithm is discussed in chapter 6.

5.2 Deriving the controllers

5.2.1 Swing-up by energy control

The swing-up of the pendulum rod can be done by a so-called energy controller. This was first done by K.Furuta and K.J.Åström. The theory behind this controller is already explained in [AsF00] and [Ber03]. The derivation of the energy based controller will be presented in this section shortly.

The swing-up of the pendulum starts with the pendulum rod in the stable downward position \( \beta = 0 \). If only the equation of motion for the pendulum rod is considered and all friction terms are neglected the equation of motion 3.2 reduces to:

\[
(J_2 + m_2 l_2^2) \ddot{\beta} + m_2 l_1 l_2 \ddot{\alpha} \cos \beta + m_2 g l_2 \sin \beta = 0
\]  

(5.1)

The dynamics of the pendulum arm will be deliberately neglected. The term \( l_1 \ddot{\alpha} \) in equation 5.1 can be seen as an input for the equation of the isolated pendulum rod. For now it is assumed that this input can take all values and can be controlled directly. This is not the case for the total system because the acceleration of the pendulum arm is dependent on its dynamics and the position of the pendulum rod. In section 5.2.3 it will be shown that, by designing a special type of controller, the acceleration of the pendulum arm can be controlled directly.
The equation of motion for the uncontrolled isolated pendulum rod has two equilibria, $\beta = 0$ and $\beta = \pi$. The total energy of the isolated pendulum is given by:

$$E = E_{kin} + E_{pot} = \frac{1}{2} (l_2 + m_3 l_2^2) \beta^2 + m_2 gl_2 (-\cos \beta - 1)$$

(5.2)

By defining the potential energy term in this way the energy of the pendulum rod in downward position is minimal and equals zero for the pendulum in upward position. With the introduction of the normalized variables $\omega_0 = \sqrt{\frac{m_2 g l_2}{(l_2 + m_3 l_2^2)}}$, $\tau = \omega_0 t$ and $\nu = \frac{l_2 \alpha}{g}$ equation 5.1 can be rewritten as:

$$\frac{\delta^2 \beta}{\delta \tau^2} + \sin \beta + \nu \cos \beta = 0$$

(5.3)

This equation is characterized by two parameters only, the natural frequency for small oscillations $\omega_0$ and the maximum acceleration of the pendulum arm which is defined as $\nu = \nu_{\text{max}} = \frac{l_2 \alpha_{\text{max}}}{g}$. The maximum acceleration of the pendulum arm is scaled with the acceleration of gravity. As can be seen from equation 5.1, the isolated pendulum is locally controllable for $\beta \neq \pm \frac{\pi}{2}$. This means that the pendulum rod can be controlled except when it is in horizontal position.

The energy based controller must somehow increase the energy of the pendulum rod from $-2m_2 gl_2$ to zero. The rate at which energy is added to the system is limited by $\nu$. To illustrate this, consider the situation as depicted in figure 5.1. The pendulum is initially in point A and has zero velocity. Now let the pendulum arm, input for the isolated pendulum, accelerate with the maximum acceleration $l_2 \alpha_{\text{max}}$. Then the gravity field seen by an observer located on the pendulum rod has direction OB, with angle $\beta_0 = \arctan(\nu)$ and magnitude $g \sqrt{1 + \nu^2}$. The pendulum rod will swing symmetrically around OB and will reach zero velocity again in point C, located at the angle $\varphi + 2\beta_0$. The energy of the pendulum is increased. Every time the pendulum rod reaches zero velocity the acceleration of the pendulum arm is reversed and energy is pumped into the system enabling the pendulum rod to reach the upright position. The energy added to the pendulum per swing is proportional to $2\beta_0 = 2 \arctan(\nu)$. 

---

Figure 5.1: Representation of the swing-up strategy
To perform energy control it is necessary to understand how the energy is influenced by the acceleration of the pendulum arm. Taking the derivative of the energy with respect to time and combining it with equation 5.1 results in:

\[
\frac{\delta E}{\delta t} = (J_0 + m_2 l_2^2) \ddot{\beta} + m_2 g l_2 \beta \sin \beta = -m_2 g l_2 \dot{\beta} \cos \beta
\]  

(5.4)

With \( u = l_1 \dot{\alpha} \) the input of the isolated pendulum rod. From equation 5.4 it is easy to see that to increase the energy of the system the input should always have the opposite sign of \( \dot{\beta} \cos \beta \). A control law is derived with the Lyapunov method. Take as a candidate Lyapunov function:

\[
V = \frac{1}{2} (E - E_0)^2
\]

(5.5)

Then for the derivative of this Lyapunov function follows:

\[
\frac{\delta V}{\delta t} = (E - E_0) \dot{E} = -m_2 g l_2 \dot{\beta} \cos \beta (E - E_0)
\]

(5.6)

The control law that stabilizes the energy equation and makes the derivative of the Lyapunov function negative definite looks like:

\[
u = k (E - E_0) \dot{\beta} \cos \beta
\]

(5.7)

With \( E \leq 0 \). Because the positions \( \beta = \pm \pi / 2 \) are not maintainable as stable positions the control law will bring the pendulum to the desired energy level \( E_0 = 0 \). To change the energy as fast as possible the control law can be changed into:

\[
u = n \text{sign} \left( (E - E_0) \dot{\beta} \cos \beta \right)
\]

(5.8)

This control law may result in chattering when solely used to control the pendulum to the upright position. Therefore the energy based control law must be used in combination with a stabilizing and balancing controller. When the pendulum is close enough to the upright position the control will be switched from the energy controller to this balancing controller, which is constructed in the next section. From equation 5.8 it follows that the input signal equals zero if the pendulum is in downward position at rest. This is mostly the initial condition and therefore the controller must be activated by some small initial input signal. Depending on the value of \( n \) the swing-up behavior of the pendulum will change speed. For \( n \geq 1 \) the pendulum can be brought up with one swing. In the total pendulum setup the control law actually computes the needed acceleration of the tip of the pendulum arm. This can be translated to a rotational acceleration of the actuated degree of freedom, \( \ddot{\alpha} \). So the energy based controller must be combined with a controller that controls the acceleration of the pendulum arm and uses the output of the swing-up controller as a setpoint. This can be an ordinary PID controller but in section 5.2.3 another controller for the actuated rotation is derived.

### 5.2.2 Linearized controller

For both the situations, the pendulum in upward and in downward position, a linearized controller is designed. A linearized model around the equilibrium point is needed for both controllers. The disadvantage of a linearized controller is the limited region of attraction. This means that the controller is not able to let the system reach its equilibrium point for all initial conditions of the pendulum rod. If the pendulum rod has a high velocity at the moment the linearized controller takes over, it is possible that it is not able to reduce the speed of the pendulum rod. The pendulum will pass the equilibrium position and the linearized region.
A control law $u = -K \vec{x}$ will be designed using pole placement techniques. The region of attraction is highly dependent on the chosen poles. The state of the model is chosen as $\vec{x} = [\alpha \quad \dot{\alpha} \quad \beta \quad \dot{\beta}]^T$ and first a linearized balancing controller for the upright position $x_{eq} = [0 \quad 0 \quad \pi \quad 0]^T$ is derived. In Appendix E.1 the linearisation is carried out. Hence the linearized model for the upright position is:

$$
\begin{bmatrix}
J_1 + m_2 g l_2^2 & -m_2 g l_2 \beta \\
-m_2 g l_2 \beta & J_2 + m_2 g l_2^2
\end{bmatrix} \begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} + \begin{bmatrix}
b_1 & 0 \\
0 & b_2
\end{bmatrix} \begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} +
\begin{bmatrix}
0 & 0 \\
0 & -m_2 g l_2
\end{bmatrix} \begin{bmatrix}
\ddot{\alpha} \\
\ddot{\beta}
\end{bmatrix} = \begin{bmatrix} K_u \\
0
\end{bmatrix} u
$$

(5.9)

This linearized system can be written in state space format like:

$$
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} = \begin{bmatrix}
p_1 p_3 - p_2^2 & 0 & 0 & 0 \\
p_1 p_2 & p_2 p_4 & -p_2 b_2 & 0 \\
0 & p_1 p_3 & -p_1 p_4 & -p_1 b_2 \\
0 & -p_2 b_1 & p_1 p_4 & -p_1 b_2
\end{bmatrix} \begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} + \frac{1}{p_1 p_3 - p_2^2} \begin{bmatrix}
0 \\
p_3 K_u \\
p_2 K_u
\end{bmatrix} u
$$

(5.10)

with:

$$
p_1 = J_1 + m_2 l_2^2 \\
p_2 = m_2 g l_2 \\
p_3 = J_2 + m_2 g l_2^2 \\
p_4 = m_2 g l_2
$$

Now the linear system matrices are known, a stabilizing controller can be derived. The existence of such a linear state feedback controller is guaranteed if the controllability matrix of the system has full rank. The controllability matrix of system 5.10 is defined as:

$$
\mathbf{P} = \begin{bmatrix}
\mathbf{B}_{up} & \mathbf{A}_{up} \mathbf{B}_{up} & \mathbf{A}_{up}^2 \mathbf{B}_{up} & \mathbf{A}_{up}^3 \mathbf{B}_{up}
\end{bmatrix}
$$

(5.11)

After evaluating this expression it is concluded that the matrix $\mathbf{P}$ has rank 4 and the system is completely controllable. Of course it is also very important to examine how close this matrix $\mathbf{P}$ is to singularity. Even if it has a rank of 4, numerical problems can arise if the matrix is almost singular. To examine this, the parameters estimated, measured and computed in chapter 4 are entered in equation 5.11. The determinant of this matrix $\mathbf{P}$ then equals $1.68 \times 10^4$, and therefore it can be concluded that this matrix is far from singular. This means that the poles of the closed loop system with feedback $u = -K \vec{x}$, and closed loop dynamics $\ddot{\vec{z}} = (\mathbf{A}_{up} - \mathbf{B}_{up} \mathbf{K}) \vec{z}$, can be placed arbitrarily by choosing $\mathbf{K}$. The resonance frequencies and phase characteristics of the system limit the pole locations though. The parameters in the matrix $\mathbf{K}$ are in fact ordinary PD controllers for $\alpha$ and $\beta$, and attention has to be given to the bandwidth and the phase margin of the open loop system. The bandwidth of the open loop system can not be chosen very high with a normal PD controller, because there exist a large phase delay for the high frequency range, see section 4.2, which can make the system unstable. Also the resonances of the system may not gain magnitude. Furthermore the differential actions in the control law amplify the measurement noise which, if chosen too high, can be a source of instability also. These considerations are important for the final design of the linearized controllers and the pole locations of the closed loop system.
CHAPTER 5. CONTROLLING THE FURUTA PENDULUM

Also for the pendulum rod in downward position, a linearized controller is designed. This is described in Appendix E.2. The linearized model for the equilibrium position \( x_{eq} = [0 \ 0 \ 0 \ 0]^T \) is:

\[
\begin{bmatrix}
J_1 + m_2 l_2^2 & m_2 l_1 l_2 \\
m_2 l_1 l_2 & J_2 + m_2 l_2^2
\end{bmatrix}
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} +
\begin{bmatrix}
b_1 \\
b_2
\end{bmatrix}
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} +
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} =
\begin{bmatrix}
K_u \\
0
\end{bmatrix} u
\]

(5.12)

And, like the model for the upright position, this model can be written in state space format like:

\[
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} = \frac{1}{p_1 p_3 - p_2^2}
\begin{bmatrix}
0 & p_1 p_3 - p_2^2 & 0 & 0 \\
0 & -p_3 b_1 & p_2 p_4 & p_2 b_2 \\
0 & 0 & 0 & p_1 p_3 - p_2^2 \\
0 & p_2 b_1 & -p_3 b_4 & p_1 b_2
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} +
\frac{1}{p_1 p_3 - p_2^2}
\begin{bmatrix}
0 \\
p_2 K_u \\
0 \\
-p_2 K_u
\end{bmatrix}
\]

With the same definitions for \( p_1, p_2, p_3, p_4 \). Also for this model the controllability matrix has full rank. When the model parameters in this matrix are replaced by their values estimated in chapter 4 it follows that the determinant equals \( 1.68 \times 10^4 \) also. Again the resonance and phase characteristics of the system limit the P and D control actions, and therefore the pole locations of the closed loop system, due to possible instability. In section 5.3 the poles are chosen and regions of attraction for both controllers will be discussed.

5.2.3 Input/Output linearizing controllers

In this section the Input/Output linearizing controllers are designed. The effect of such a controller is that it renders the system input to output behavior in a linear way. To be able to derive the controller, a few instruments are needed. These will be discussed first with the use of a standard SISO system of the form:

\[
\begin{align*}
\dot{x} &= f(x) + g(x)u \\
y &= h(x)
\end{align*}
\]

(5.14)

The relative degree of a system is defined as the number of successive differentiations of the output of the system, \( y = h(x) \), for which the input, \( u \), is explicitly present for the first time. In other words the relative degree, \( r \), can be described as:

\[
y^r = f_s(x) + f_u(x)u, \quad \text{with } f_u(x) \neq 0
\]

(5.15)

With the definition of the Lie derivative

\[
L_{f_1} f_2 = \frac{\partial f_2}{\partial x} f_1
\]

(5.16)

\[
L_{f_1} f_2 = L_{f_1}^{-1} L_{f_1} f_2 = L_{f_1}^{-1} \left( \frac{\partial f_2}{\partial x} f_1 \right)
\]

(5.17)

equation 5.15 can also be written as:

\[
y^r = L_f h(x) + L_{g_i} L_{f_i}^{-1} h(x) u \\
L_g L_{f_i}^{-1} h(x) \equiv 0 \quad i = 0, 1, \ldots, r - 2
\]

(5.18)

\[
L_g L_{f_i}^{-1} h(x) \neq 0
\]
CHAPTER 5. CONTROLLING THE FURUTA PENDULUM

For a SISO system in general, the requirement to be Input/Output exact linearizable is that its relative degree must equal the order of the system. When the relative degree of a system is less, it is said to be Input/Output linearizable. The control law following from equation 5.18 that linearizes the input to output behavior of the system, given by equation 5.14 and with relative degree \( r \), is given by:

\[
    u = \frac{1}{L_f L_f^{-1} h(x)} (-L_f h(x) + v)
\]  

(5.19)

In this equation \( v \) is the new input, and because now \( y' \) equals \( v \) the output of the system can be controlled in a linear way. This is the basic idea of Input/Output linearizing controllers. The Furuta pendulum can be seen as a single input, multi output system. To make it a SISO system, which in this case is needed to design an Input/Output linearizing controller, one of the two degrees of freedom has to be neglected. Two different controllers are designed, one for each rotation. To convert the pendulum model to SISO the output of the system has to be chosen as \( y = \alpha \) or \( y = \beta \). By neglecting one degree of freedom the assessment of the stability of the complete system is more difficult. To be able to examine the stability the system is written in so-called normal form. This is done by using the coordinate transformation:

\[
    \Phi : x \mapsto \begin{pmatrix} h(x) \\ L_f h(x) \\ \vdots \\ L_f^{r-1} h(x) \\ \eta \end{pmatrix}
\]  

(5.20)

Where \( \eta \) consists of \( n - r \) functions that satisfy \( d\eta(x)g(x) = 0 \). We refer to these \( r \) new variables \( h(x), L_f h(x), \ldots, L_f^{r-1} h(x) \) as \( \xi \). The new system equations then read:

\[
    \begin{align*}
        \dot{\xi}_1 &= \xi_2 \\
        \dot{\xi}_2 &= \xi_3 \\
        &\vdots \\
        \dot{\xi}_r &= v \\
        \dot{\eta} &= a(\xi, \eta) \\
        y &= \xi_1
    \end{align*}
\]  

(5.21)

The zero dynamics of this new system are defined by \( \dot{\eta} = a(0, \eta) \). They represent the dynamics that remain, after the controlled output has reached a constant setpoint and the tracking error is zero. It can be proven that if the zero dynamics of this system are stable, the original system is also stable.

In this section a few tools used to design an I/O linearizing controller have been discussed briefly. For proofs and more information about relative degree, normal form and zero dynamics see [Sas99].
CHAPTER 5. CONTROLLING THE FURUTA PENDULUM

Input/Output linearizing controller for \( \alpha \)

To derive a linearizing controller we start with the equations of motion as given in equation 3.2, but slightly rearranged and assuming Coulomb friction compensation:

\[
\begin{align*}
\dot{\mathbf{q}} &= \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix}, \\
\mathbf{g} &= \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix} = \begin{bmatrix} M^{-1} \left( \mathbf{f} \right) \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \mathbf{f} \end{bmatrix} \mathbf{u} \\
&= f(\mathbf{q}) + gu \\
y &= \alpha = h(\mathbf{x})
\end{align*}
\]

(5.22)

(5.23)

with:

\[
\begin{align*}
\mathbf{M} &= \begin{bmatrix} J_1 + m_2 l_1^2 + m_3 l_2^2 \sin^2 \beta & m_2 l_1 l_2 \cos \beta \\ m_2 l_1 l_2 \cos \beta & J_2 + m_2 l_2^2 \end{bmatrix} \\
\mathbf{D} &= \begin{bmatrix} b_1 + m_2 l_2^2 \sin \beta \cos \beta & m_2 l_2 \dot{\alpha} \sin \beta \cos \beta - m_2 l_1 l_2 \dot{\beta} \sin \beta \\ -m_2 l_2 \dot{\alpha} \sin \beta \cos \beta & b_2 \end{bmatrix} \\
\mathbf{F} &= \begin{bmatrix} 0 \\ m_2 g l_2 \sin \beta \end{bmatrix} \\
f &= \begin{bmatrix} K_u \\ 0 \end{bmatrix}
\end{align*}
\]

The relative degree of this system equals 2, see Appendix F.1. So the system is not exact I/O linearizable. This means that not all states can be controlled at the same time, which could be expected for a underactuated system. From the preceding section the Input/Output linearizing controller is given by:

\[
u = \frac{1}{L_p L_f^{-1} h(\mathbf{x})} (-L_f h(\mathbf{x}) + v)
\]

(5.24)

When evaluating this control law, which is done in Appendix F.1, it becomes clear that this expression is very complex. The control law is computed using the symbolic toolbox of Matlab and is not written down explicitly. Now the control law is known, the input to output behavior of the Furuta pendulum system is given by:

\[
\ddot{y} = \ddot{\alpha} = v
\]

(5.25)

In most cases the output of the system is the one that needs to be controlled and not its second derivative. Assume that the controlled output has to follow a certain setpoint or has to reach a constant value. A position setpoint can be translated to an acceleration setpoint by taking its second derivative. In the ideal case, no unmodeled dynamics or measurement noise, this acceleration setpoint would be sufficient to bring the pendulum rod to its desired position. But because some assumptions are made during the modeling process this is not the case. Moreover, because of the double integrator behavior and an initial error, which is always present due to numerical errors, the position error will grow unboundedly. To overcome this problem in some extent the control law for \( v \) is chosen as:

\[
v = \ddot{\alpha} = -P (\alpha - \alpha_m) - D (\dot{\alpha} - \dot{\alpha}_m) + \ddot{\alpha}_m \Rightarrow \ddot{\alpha} = D \ddot{\alpha} + P \dot{\alpha}_m = 0
\]

(5.26)

In which the subscript 'm' refers to the characteristics of the setpoint. Of course the stability is dependent on the choice of the parameters \( P \) and \( D \) and the influence of unmodeled dynamics. With this control law for \( v \), the design of the I/O controller for the actuated rotation \( \alpha \) is complete. The control law is given by equations 5.24 and 5.26. Once again, because of the complexity of this expression, it is not written down explicitly. The implementation is presented in section 5.3.
To examine the stability of the total system, the zero dynamics have to be evaluated. From Appendix F.1, the zero dynamics of the system with the I/O linearizing control law read:

\[
\begin{aligned}
\dot{\alpha} &= 0 \\
\ddot{\alpha} &= 0 \\
\dot{\beta} &= \ddot{\beta} \\
\ddot{\beta} &= -m_2g l_2 \sin \beta - b_2 \dot{\beta}
\end{aligned}
\]  

(5.27)

It can be seen that these zero dynamics are stable and, in fact, represent the free, damped response of the pendulum rod to gravity. The pendulum rod will eventually reach the stable equilibrium point, \( \beta = 0 \).

**Input/Output linearizing controller for \( \beta \)**

In order to derive a I/O linearizing controller for the free rotation, \( \beta \), the same equations of motion are used as in the preceding section, see equation 5.22. Because now the rotation of the pendulum rod is to be controlled, the output equation becomes:

\[ y = \beta = h(x) \]  

(5.28)

In Appendix F.2, it is computed that the relative degree for this system equals 2, so again the controller can not control all states. Again from section 5.2.3, the control law is given by:

\[ u = \frac{1}{L_\beta L_f^{-1} h(x)} (-L_f h(x) + v) \]  

(5.29)

In Appendix F.2, this control law is evaluated and it is concluded to be too complex to write down. After implementing the control law, the input to output behavior of the closed loop system becomes:

\[ \dot{y} = \ddot{\beta} = v \]  

(5.30)

And as is done for the I/O controller for the actuated rotation, the control law for \( v \) is chosen as:

\[ v = \ddot{\beta} = -P(\beta - \beta_m) - D(\dot{\beta} - \dot{\beta}_m) + \ddot{\beta}_m \Rightarrow \ddot{\beta} + D\dot{\beta} + P\beta = 0 \]  

(5.31)

With equations 5.29 and 5.31, the design of the I/O linearizing controller for the free rotation is finished. In the next section, the results of the simulations and experiments with this controller are discussed.

From Appendix F.2, it follows that the zero dynamics of the total system are given by:

\[
\begin{aligned}
\dot{\alpha} &= \dot{\alpha} \\
\ddot{\alpha} &= 0 \\
\dot{\beta} &= 0 \\
\ddot{\beta} &= 0
\end{aligned}
\]

(5.32)

From this it can be concluded that, after \( \beta \) has reached a constant value, the system is marginally stable. The angular velocity of the actuated rotation, \( \dot{\alpha} \), does not decrease or increase but remains constant. How this affects the dynamics of the total system and its controllability will be discussed in the next section and in Appendix G.
CHAPTER 5. CONTROLLING THE FURUTA PENDULUM

5.3 Simulation and experimental results

This section deals with some simulations and experiments performed with the controllers, or combinations of them, designed in the preceding sections. The Swing-up controller, a combination between the energy based controller and the I/O linearizing controller for $\alpha$, the I/O linearizing controller for $\beta$ and the linearized controller for $\beta = 0$ are treated in Appendix G. Only two special cases are discussed in this section. The first one is the Swing-up mode. In this mode the pendulum rod is brought close to the upward position by the Swing-up controller, and subsequently a linearized controller takes over to balance it in upright position. The second one is the Swing-down mode by I/O control. For this mode a special setpoint has to be generated to assure fast stabilization of the pendulum rod in downward position.

![Figure 5.2: Animation used to visualize responses of the pendulum](image)

To visualize the response of the Furuta pendulum during simulations, an animation S-function in '.m' is written first. By looking at the animation and the response of the pendulum in total it is possible to better interpret this response and actually see what happens. Instead of looking at the responses of the four degrees of freedom individually. In figure 5.2 a snapshot of the visualization is presented. This animation runs together with the simulation. The implementation of the Furuta pendulum model is discussed in chapter 3. To model the output limits of the TUE DACS, and thus the amplifier and motor torque, a saturation is used. In this way the model of the Furuta pendulum used during simulations and the real pendulum used during experiments have exactly the same input and outputs. For implementation see Appendix D. The two blocks can easily be interchanged and nothing in the Simulink model has to be altered in order to switch from simulation to experiment. Eventually all individual 'pendulum modes' the Furuta pendulum can perform will be combined to create a demonstration experiment. The combination of, and switching between all different controllers and modes will be described in the chapter 6. To be able to couple all different modes they have to be compatible. After every mode the pendulum must have a final state which enables the next mode and controller to perform its task. For some modes some extra precautions must be made in order to achieve this. This also will be discussed in this section and in Appendix G.
5.3.1 Swing-up controller + linearized controller

Implementation

When the Swing-up controller, treated in Appendix G.1, is combined with the linearized controller, around the unstable equilibrium point $\beta = \pi$, and takes over when the pendulum is near this point, stabilization should be possible. This combination, of three controllers, will be referred to as 'Balancing controller'. To implement the linearized controller, simple standard Simulink blocks are used. To activate the 'Balancing mode' and at the same time deactivate the 'Swing-up mode', a switching strategy has to be designed and implemented. The strategy determines when the pendulum rod is close enough to its upright position and when the linearized controller can take over. The implementation of this switching algorithm is given in chapter 6.

Simulations and experiments

The region of attraction of the linearized controller plays an important role, because it determines the limits for the switching algorithm. After a few simulations the switching point from Swing-up to Balancing mode is chosen as $\pm 30^\circ$ relative to the upright position. To prevent chattering of the controller when the pendulum rod is close to this $30^\circ$ the switching point from Balancing to Swing-up mode is placed lower, at $\pm 35^\circ$. Next, the poles of the closed loop system $z = (A_{\text{op}} - B_{\text{up}}K)z$, have to be chosen. They can be chosen arbitrarily but every pole has its effect on the region of attraction for the pendulum rod. A few basic thoughts, that formed the fundament for the choice of the poles, are:

- The pole for $\beta$ has to be relatively large in order to be able to attract the pendulum rod fast enough to its upright position.
- The pole for $\dot{\beta}$ has to be relatively large to make sure the linearized controller is able to reduce the high speed of the rod when the linearized controller is enabled.
- The poles for $\alpha$ and $\dot{\alpha}$ have to be relatively small in order to enable the controller to fulfill the preceding requirements. For example the controller may not react too much on an error in $\alpha$, which can become rather high, because otherwise it would slam the pendulum rod away from its upright position and linear region.

With these considerations and the limitations with respect to stability requirements, discussed in section 5.2.2, in mind and after a few small experiments the poles of the closed loop system have been chosen as $P_{\text{up}} = [-0.1 -1.5 -15 -20]$. A way to determine the bandwidth of the open loop system from input to $\beta$, which is not done in this report, is to determine the FRF of the pendulum rod when it is balanced in upward position. Subsequently apply the PD controller that follows from the pole locations to this measured FRF. The bandwidth and phase margin can be determined from this FRF.

It does not matter were the pendulum arm reaches zero velocity during this mode and it does not necessarily have to be the starting position of the pendulum arm. The linearized controller is made independent of the error in $\alpha$ simply by setting it equal to zero. For the poles mentioned and the $\alpha$-error set to zero the results of the simulation are given in figure 5.3. From these results it can be concluded that the linearized controller is able to stabilize the pendulum rod fast. Especially for high values of $n$, the pendulum rod can be brought up fast and can stabilized within 1.5 seconds. The balancing controller needs approximately 1.8 seconds to stabilize the pendulum arm as well.

During the experiments the same pole locations are used for the closed loop linear system. The results of the experiments are also given in figure 5.3 and show a lot of similarity. One difference during swing-up is explained in the section G.1, less energy increase per swing. Although not represented in the figure, it is possible that the stabilization point differs by $2\pi$ rad. This is due to the fact that the first time the rod comes in the linear range, the linearized controller is immediately enabled. This can happen during a 'positive' or 'negative' swing.
5.3.2 I/O linearizing controller $\beta$ + linearized controller

Implementation

The I/O linearizing controller for $\beta$ is not capable of controlling the rotation of the pendulum arm. When it is used to bring the pendulum rod in upward or downward position a combination between this controller and the linearized controller is used to stabilize the complete system.

For the swing-up case the final velocity of the pendulum arm can be stabilized to zero. This is done by replacing the Swing-up controller as discussed in Appendix 5.3.1 by the I/O linearizing controller for $\beta$. The switching mechanism remains unchanged and operates in the same way as it does for the Balancing controller. This means that it switches from Swing-up mode, now executed by the I/O controller, to balancing mode for pendulum rod angles $\pm 30^\circ$ relative to the upright position. The closed loop poles of the linearized model remain the same. From Appendix G.3 it can be learned that for the balancing mode in downward position it is almost impossible to control the pendulum arm itself. Moreover, after a few simulations it could be concluded that the initial pendulum arm velocity for the linearized controller is very important. If this velocity is too high the pendulum rod can not be stabilized. Something has to be found to make the initial conditions of the Furuta pendulum state, when switching to the linearized controller around $\beta = 0$, as suitable as possible. This comes down to minimizing the pendulum arm velocity at the moment of switching. The closed loop poles for the linear model are not changed.

For both cases, swing-up and swing-down, the I/O linearizing controller works parallel with a linearized controller, and control will be handed over to one of them by the switching algorithm.
Simulations and experiments

The combination of the I/O linearizing controller and linearized controller for the upright position works in exactly the same way as the Balancing controller with energy control. There is not a great difference between the two setups except for the fact that the combination with the I/O controller is capable of swinging up the pendulum rod a little bit faster. This case will not be discussed any further.

The other case though, is more difficult. Because the linearized controller cannot control the pendulum arm velocity, some precautions must be made to make sure the final pendulum arm velocity is not too high for successive pendulum modes. When the pendulum rod swings down, controlled by the I/O linearized controller, the moment of switching to the linearized controller is very important. Its imaginable that the linearized controller will accelerate the pendulum arm to decelerate the rod. The final pendulum arm velocity, \( \dot{\alpha}_m \), is approximately equal to the initial velocity at the moment of switching added with the integral over the acceleration profile. The acceleration profile is mostly dependent on the pendulum rod velocity. The pendulum arm velocity should be as low as possible at the switching point. In order to determine when to switch, or what can be adjusted to lower the pendulum arm velocity, we look at the setpoint profile for the Swing-down mode, treated in Appendix G.2, and compare it with a 'normal' swing-down profile. From figure 5.4 it can be concluded that the pendulum arm velocity reaches a local minimum around the top of each small swing-up in the setpoint profile and therefore that is the best point to switch to the linearized controller. The results for this switching strategy are presented in figure 5.5. The final pendulum arm velocity in this setup is 2.9 rad/s during the simulation and approximately 0 rad/s for the experiment. The successive pendulum mode can be executed under this conditions.

![Figure 5.4: Two swing-down setpoint profiles](image-url)
Figure 5.5: Results for the swing-down mode in combination with the balancing controller
Chapter 6

Controller switching

6.1 Introduction

In the preceding chapter a few different kind of pendulum modes have been discussed. The available pendulum modes are:

- Swing-up by energy control (*)
- Swing-up by energy control and balance with linearized controller
- Free-fall and balance down with linearized controller
- Swing-up by I/O linearisation (*)
- Swing-up by I/O linearisation and balance up with linearized controller
- Swing-down by I/O linearisation (*)
- Swing-down by I/O linearisation and balance down with linearized controller
- Swing 360 degrees by I/O linearisation (*)
- Shake by I/O linearisation
- No controller

And for all cases with the I/O linearisation the mode is able to execute in clockwise or counterclockwise direction. The shake mode has two variants also, the 'fast' shake and the 'slow' shake. In total there are 16 different modes the pendulum can perform. To be able to execute them successively they must be compatible with each other. For each mode of the Furuta pendulum the initial conditions have to preserve the stability of the controller-pendulum system. This is already discussed in the preceding chapter. All modes that can be executed, except the ones with (*), have final conditions enabling the next mode to execute without destabilizing the system. The modes with (*) have a final pendulum arm speed that does not guarantee the next mode to execute correctly. Another requirement for compatibility is that each pendulum mode must be able to execute correctly, regardless of all preceding and already executed pendulum modes. To obtain this, some algorithm or mechanism has to be designed that is able to couple all the pendulum modes. Sometimes a pendulum mode itself consists of two or more stages, like the Swing-up and balancing mode. In that case, control has to be switched within a pendulum mode. The switching strategy for these multiple-stage modes is already discussed in the preceding chapter. But also for this case some switching mechanism has to be implemented. It is decided to integrate the switching mechanism for the modes and stages. The switching mechanism, that mostly consists of logical operators, is programmed as a S-function in C-language. The total switching algorithm

29
is given in Appendix H. The switching mechanism has 4 inputs and 13 outputs. The signals of the linearized controller, both I/O controllers and the actual pendulum state make the inputs. The output consists of a computed motor torque and 12 outputs to enable and disable the correct modes. The total Simulink model with this switching mechanism is presented in Appendix D.

### 6.2 Switching between modes

#### Initializing the pendulum modes

Some of the Furuta pendulum modes are dependent on the already performed modes. Especially the modes that are executed using some kind of setpoint, the modes with the I/O controller for $\beta$, need the initial state of the Furuta pendulum. That is, the setpoint is dependent on this initial state. This setpoint is dependent on the pendulum rod angle $\beta$ at the start of the mode. It can already have made multiple revolutions in clock or counterclockwise direction. Because the pendulum rod will always be stabilized in upward or downward position the initial angle differs by multiples of $\pi$. Every time the mode is executed the setpoint can be different. All generated setpoints for the swing modes have initial position zero and are always defined counterclockwise. To get a clockwise swing the position, velocity and acceleration setpoint are multiplied with $-1$.

Dependent on the initial angle $\beta$ the position setpoint is also added with the initial multiple of $T$, to form the final setpoint. Every time a new mode is executed the function NextMode, programmed in the Switching algorithm, is called first, and is programmed as:

```c
static void NextMode(SimStruct *S)
{
    InputRealPtrsType uPtrs = simGetInputPortRealSignalPtrs(S, 0);
    real_T *beta_start = simGetInputPortRealSignal(S, 12);
    real_T *t = simGetTPtr(S);

    extern int mode_number;
    extern real_T t_start;

    mode_number += 1;
    t_start = *t;

    if (fabs(beta - M_PI + ceil(beta / M_PI)) <= fabs(beta - M_PI + floor(beta / M_PI))){
        *beta_start = M_PI + ceil(beta / M_PI);
    } else if (fabs(beta - M_PI + ceil(beta / M_PI)) >fabs(beta - M_PI + floor(beta / M_PI))){
        *beta_start = M_PI + floor(beta / M_PI);
    }
}
```

The function computes, besides the starting time of each mode, the initial angle $\beta$, in multiples of $\pi$. This starting angle is outputted by the S-function and is added to the general setpoint in the Simulink model. In this way the setpoint is always compatible with the actual state of the Furuta pendulum and the mode can be executed.

The modes that use the energy based controller and the I/O controller for $\alpha$ are initialized in a slightly different way. Now the acceleration of the pendulum arm is computed and the position setpoint is obtained by integrating this signal twice. Every time the energy based Swing-up mode is initialized the switching mechanism triggers the integrator block in Simulink by giving a pulse. Then the initial condition of the second integrator block, is set to the actual pendulum arm angle $\alpha$.

For the modes that use one of the two linearized controllers, again another strategy is used. The linearized controllers need the actual state of the Furuta pendulum with respect to the equilibrium position. Only the actual state of $\beta$ is needed though, because $\alpha$ may take any value and $\dot{\alpha}$ and $\ddot{\alpha}$
are zero in the equilibrium position. Before the linearized controller is activated the pendulum rod may have made multiple revolutions. When control is handed over to the linearized controller, the actual angle of the pendulum rod must be normalized in order to express it relatively to the equilibrium position $\beta = \pi$ or $\beta = 0$. This is done by unwrapping the actual pendulum rod angle to an angle expressed in the range $[-\frac{\pi}{2}, \frac{3\pi}{2}]$. In this way the angle is always expressed correctly for the upward and downward position. The unwrapping takes place in Simulink and not in the switching mechanism itself.

**Executing the pendulum modes**

When a pendulum mode is initialized correctly, it can be executed. All controllers generate output simultaneously during all modes and all controller signals are send to the switching mechanism. Dependent on the currently running pendulum mode and stage, the switching algorithm outputs the right controller signal to the Furuta pendulum robot. When the execution time is over, the next mode will be initialized.

The modes that use a setpoint for $\beta$, thus the modes that use the I/O controller for $\beta$, are enabled by the switching algorithm by means of sending a 'high' signal to the matching enable port of the setpoint generator. When the mode has reached its execution time, the setpoint generator is disabled again by holding a 'low' signal. All setpoint generators for the I/O controller for $\beta$ are summed first and subsequently send to the controller. Because every setpoint generator is reset and set to zero when it is disabled, the different setpoint generators will never interfere with each other if only one generator is enabled at the same time.

The modes using the energy based swing-up controller are enabled by sending a 'high' signal to the enable port of the matching block in Simulink. The output of this controller is send to the I/O controller for $\alpha$. Meanwhile the setpoint for this controller has been reset to match the actual pendulum arm state, and a compatible control signal is send to the switching mechanism.

### 6.3 Switching between stages

The pendulum modes that consist of two stages, combinations of Swing-up or Swing-down and a linearized controller, have to be enabled and disabled within the same mode. Dependent on the actual state of the Furuta pendulum a stage has to be enabled or disabled. This has to be implemented in the switching mechanism. For the energy based Swing-up mode the code is programmed as:

```c
else if (sequence[mode_number].name == "Swing up by energy and balance with linearized controller"){
    if (fabs(beta) <= \pi \&\& fabs(beta_dot) <= \pi) {
        *motor_torque = 70;
    } else if (cos(beta) <= cos(SWITCH_SW_TO_BAL)) {
        *motor_torque = balance_cont;
        *balance_switch = 1;
        *swing_up_enabled = 0;
        if (((t - t_start) > sequence[mode_number].options[0]) {
            NextMode(S);
        }
    } else if (cos(beta) >= cos(SWITCH_BAL_TO_SWU)) {
        *motor_torque = IO_alpha_cont;
        *swing_up_enabled = 1;
        *swing_up_speed = sequence[mode_number].options[1];
    }
}
```

For the other modes consisting of multiple stages comparable code is written. A 'low' or 'high' signal controls the linearized controller to be in upward or downward configuration.
Chapter 7

Manual for the Furuta pendulum

In chapters 2 to 6 all information about the Furuta pendulum is presented. A model is derived in chapter 3, parameters are estimated in chapter 4 and controllers are designed in chapter 5. However the preceding chapters form a good basis for understanding and getting the Furuta pendulum operational, a few extra information for first-time users will be discussed in this chapter.

7.1 Files and folders

The home directory for the Furuta pendulum is D:\Furuta_pendulum, and in the subdirectory Furuta_pendulum_demo all files needed to get the Furuta pendulum operational are present. These files, among others, are also copied to an CD-ROM added in Appendix I. In table 7.1 all files in the directory D:\Furuta_pendulum_demo are presented and discussed.

<table>
<thead>
<tr>
<th>File</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Furuta_pendulum_model.m</td>
<td>A m-file S-function containing the model of the Furuta pendulum presented in chapter 3, assuming friction compensation. It is arguments are all model parameters and the two initial conditions ( \alpha_0 ) and ( \beta_0 ).</td>
</tr>
<tr>
<td>Furuta_pendulum_init.m</td>
<td>A m-file in which all model parameters, see chapter 4, and initial conditions are initialized. Also the two linearized models and accompanying control laws are computed in this file. And all data files needed to create the setpoints are loaded. This file runs before every simulation and experiment.</td>
</tr>
<tr>
<td>Furuta_pendulum_anim.m</td>
<td>A m-file S-function which is responsible for the animation of the Furuta pendulum during simulations. Its arguments are 2 dimensional parameters ( l_1 ) and ( l_{Rot2} ), the initial conditions ( \alpha_0 ) and ( \beta_0 ) and a drawing parameter. If this parameter is set to 1 the path of tip of the pendulum rod is drawn.</td>
</tr>
</tbody>
</table>
CHAPTER 7. MANUAL FOR THE FURUTA PENDULUM

### Table 7.1: Files in the home-directory of the Furuta pendulum

<table>
<thead>
<tr>
<th>File</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0_cont_alpha.c</td>
<td>Two C-file S-functions containing the I/O linearizing controllers for $\alpha$ and $\beta$ respectively. Their arguments are the P and D actions for the control law, see chapter 5, and they are initialized in Furuta_pendulum_init.m.</td>
</tr>
<tr>
<td>I0_cont_alpha.dll</td>
<td></td>
</tr>
<tr>
<td>I0_cont_beta.c</td>
<td></td>
</tr>
<tr>
<td>I0_cont_beta.dll</td>
<td>The compiled versions of the C-files, needed during simulations and experiments in Simulink.</td>
</tr>
<tr>
<td>Switching_algorithm.c</td>
<td>A C-file S-function, and its compiled version, containing the switching algorithm needed to switch between the controller. This file is given in Appendix II.</td>
</tr>
<tr>
<td>Switching_algorithm.dll</td>
<td></td>
</tr>
<tr>
<td>sequence.h</td>
<td>A file programmed in C-language containing the list of successive modes the Furuta pendulum has to perform. This file can be changed in order to create your own demo file. It contains all possible modes with default parameters as comments. This file is presented in Appendix II.</td>
</tr>
<tr>
<td>Template_sim.mdl</td>
<td>A Simulink model with the Furuta pendulum model and the animation file. Represents the total Furuta pendulum during simulations and can be copied into a user defined Simulink model to execute simulations.</td>
</tr>
<tr>
<td>Template_exp.mdl</td>
<td>A Simulink model containing the coupling of the real Furuta pendulum and the computer. It is complete with TUeDACs, Butterworth filters, saturation and constant transfers and ready to be copied in every experiment to be executed with the Furuta pendulum. Friction compensation is not implemented.</td>
</tr>
<tr>
<td>Furuta_control_sim.mdl</td>
<td>The two Simulink models, one for simulations and one for experiments, that are finally used to execute the demo file. They contain the switching mechanism and all controller modes. The experimental model is given in Appendix D.</td>
</tr>
<tr>
<td>Furuta_control_exp.mdl</td>
<td></td>
</tr>
</tbody>
</table>

Further the directory contains four '*.mat-files', four '*r3g-files' and 'ref3.c'. All these files are needed to generate the various setpoints for the pendulum modes, using the 3rd order setpoint generator.

### 7.2 Starting up the demonstration experiment

To be able to build your own demo file a '.h' file is written that will be included in the C-program of the switching mechanism. In this file, which is given in Appendix II, all possible modes can be listed and the pendulum will execute these modes successively. Every mode in the list is called with:

```
"Mode Name", { Execution time , Optional parameter }
```

At the bottom of this file the default parameters for each mode are added as a comment. It is important to realize that if, for example, the default execution time is lowered, the mode may not finish within this time and system can become unstable. Whenever a self-made list is entered it is important that the list is correct, so never execute two Swing-up modes successively and make sure that whenever a Swing-down mode is in the list, the pendulum is initially in upward position. For the modes without stabilization, I/O controller modes for $\beta$ without the linearized controller, extra care has to be taken because the pendulum arm velocity may be too high for the next mode.
to execute correctly. It is always wise to run a simulation with the new demo file first. An example of a demonstration experiment consisting of four different modes is:

```c
struct mode {
    char *name;
    real_t options[2];
} sequence[] = {
    "Swing up by energy and balance with linearized controller", (2, 1.5),
    "Swing down by I/O c-wise and balance with linearized controller", (0.3, 0),
    "No controller", (0.5, 0),
    "Swing up by energy and balance with linearized controller", (1.5, 1.4),
};
```

This demonstration experiment first executes the mode 'Swing-up by energy and balance with linearized controller' for 2 seconds. The optional parameter in this case is set to 1.5 and corresponds with the energy increase per swing, \( n \). The combination of successive modes can also be used to create new 'modes'. Actually, the combination of the last three modes in the list represent a turn of the pendulum rod over 360° in clock-wise direction. The advantage of this 'new mode' is that the rod will be balanced in upright position again. This is not the case in the 360° mode based on I/O control only. So by using the available modes and execution times, and combine them in a smart way, even more 'modes' can be performed.

To start the demonstration experiment first the file `Furuta-control-exp.mdl` has to be opened. Then the model needs to build to generate a real-time executable. The experiment can be started by using the standard TUEDACs commands. The mode list file `sequence.h` and the switching mechanism will be compiled automatically before the model is build. The compile commands are programmed in the initialization file that runs before every simulation and every time an experiment is build. In fact, the only file that needs to be opened to alter the demonstration experiment is the file with the mode list. All other files need not to be accessed. Once again be careful with the choice of the sequence of the pendulum modes in the list. Make sure it is compatible and be aware of the fact that the modes without stabilization may prevent the successive mode to complete or execute. First verify your demonstration experiment using by simulation.

### 7.3 Pendulum data

In this report a lot of important information and data about the experimental Furuta pendulum is presented. To be able to find the desired parameter needed for modeling or other purposes quickly, the most relevant Furuta pendulum data is gathered in table 7.2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_1 )</td>
<td>m</td>
<td>Length of pendulum arm</td>
<td>0.25</td>
</tr>
<tr>
<td>( l_2 )</td>
<td>m</td>
<td>Length of axis of rotation to center of mass of pendulum rod</td>
<td>0.265</td>
</tr>
<tr>
<td>( l_{rod} )</td>
<td>m</td>
<td>Total length of pendulum rod</td>
<td>0.61</td>
</tr>
<tr>
<td>( m_2 )</td>
<td>kg</td>
<td>Mass of pendulum rod</td>
<td>0.298</td>
</tr>
<tr>
<td>( J_1 )</td>
<td>kgm²</td>
<td>Moment of inertia of pendulum arm</td>
<td>0.36</td>
</tr>
<tr>
<td>( J_2 )</td>
<td>kgm²</td>
<td>Moment of inertia of pendulum rod</td>
<td>0.0064</td>
</tr>
<tr>
<td>( h_1 )</td>
<td>Nms/rad</td>
<td>Viscous friction for pendulum arm</td>
<td>0.68</td>
</tr>
<tr>
<td>( h_2 )</td>
<td>Nms/rad</td>
<td>Viscous friction for pendulum rod (assumption)</td>
<td>0.0001</td>
</tr>
<tr>
<td>( C_1 )</td>
<td>Nm</td>
<td>Coulomb friction in pendulum arm</td>
<td>4.0</td>
</tr>
<tr>
<td>( T_m )</td>
<td>Nm/V</td>
<td>Transfer from TUEDACs to motor torque</td>
<td>64</td>
</tr>
<tr>
<td>( n_m )</td>
<td>-</td>
<td>Number of counts/rev. of the pendulum arm (interpolation factor 1)</td>
<td>163840</td>
</tr>
<tr>
<td>( n_\theta )</td>
<td>-</td>
<td>Number of counts/rev. of the pendulum rod</td>
<td>4000</td>
</tr>
</tbody>
</table>

Table 7.2: Most relevant pendulum data
Chapter 8

Conclusions and recommendations

During this internship the Furuta pendulum is revitalized. First of all, a new and faster computer is connected to the robot. Also the acquisition system is changed from a DSpace board to the TUeDACS system. After describing the behavior of the Furuta pendulum with a mathematical model using Lagrange’s equations, the model parameters are estimated with a SMC. All parameters that could be measured directly are not estimated. The estimated parameters differ considerably from previous results, most likely due to the new setup of the robot. The model parameters have been verified to be expressed in SI units by performing a differential measurement. Also a very simple friction compensation is derived to overcome the main part of the non-linear friction behavior. After the estimation procedure the controllers for the Furuta pendulum are designed. In total three different types of controllers, an energy controller, two linearized controllers and two I/O linearizing controllers, are designed. The latter type is of such complexity that it can not be modeled easily with standard Simulink blocks and the use of the C programming language is needed. Combinations of the controllers are needed to stabilize the pendulum rod in upright or downward positions. It can be concluded that especially the stabilization of the downward position is hard, with respect to the upright position, due to the direction of gravity. All controllers, and combinations between them, are tested during simulations and experiments. To visualize the response of the robot during simulations and to better understand its behavior, animation file is programmed. The robot is able to stabilize the pendulum rod, in upward or downward position, within 2 seconds. The Furuta pendulum is able of performing 16 different modes, of which 9 are globally asymptotically stable and 7 are marginally stable. During an experiment the stable modes can be placed in arbitrary order to create a user defined demonstration. To switch between the different pendulum modes a switching algorithm is designed and implemented in C-code. A manual is written, in the form of this report, and the Furuta pendulum now is ready for future students.

One of the improvements for the control of the Furuta pendulum is the implementation of a more complex friction compensation. Especially when the pendulum arm velocity is small, the used compensation during this internship actually does not satisfy. And this resulted in the fact that it is very hard to control the pendulum arm velocity exactly to zero. This is also due to the position dependency of the friction in the pendulum arm. Various, more complex, friction compensations have already been designed for the pendulum. Due to the new setup though, these compensations can not be used directly. Also other control strategies, like fuzzy control, have already been designed for this robot. It is recommended to try and translate the previous results with the Furuta pendulum to the new setup.

Now the demonstration experiment is finished, it is a great way to show the possibilities of control and of the Furuta pendulum. Especially during open house of the university or other public days, this experiment is a nice demonstration.
Bibliography


Appendix A

Datasheets

A.1 Heidenhain ERN 480 rotational encoder

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter</td>
<td>58 mm</td>
</tr>
<tr>
<td>Incremental signals</td>
<td>1 Vpp</td>
</tr>
<tr>
<td>Line counts</td>
<td>5000</td>
</tr>
<tr>
<td>Cutoff frequency (-3 dB)</td>
<td>&gt; 180 kHz typical</td>
</tr>
<tr>
<td>Cutoff frequency (-6 dB)</td>
<td>&gt; 450 kHz typical</td>
</tr>
<tr>
<td>Power supply</td>
<td>5 V ± 10%</td>
</tr>
<tr>
<td>Max. current consumption (without load)</td>
<td>120 mA</td>
</tr>
<tr>
<td>Mechanical permitted speed n</td>
<td>max. 12000 rpm</td>
</tr>
<tr>
<td>Starting torque</td>
<td>&lt; 0.01 Nm</td>
</tr>
<tr>
<td>Inside diameter</td>
<td>12 mm</td>
</tr>
<tr>
<td>Vibration (55 - 2000 Hz)</td>
<td>300 m/s²</td>
</tr>
<tr>
<td>Max. operating temp.</td>
<td>100 °C</td>
</tr>
<tr>
<td>Weight</td>
<td>0.25 kg</td>
</tr>
</tbody>
</table>

A.2 The Dynasyn DV7-6-4M AC motor

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_n )</td>
<td>0.68 kW</td>
</tr>
<tr>
<td>( M_n )</td>
<td>6.4 Nm</td>
</tr>
<tr>
<td>( U_n )</td>
<td>190 V</td>
</tr>
<tr>
<td>( I_n )</td>
<td>3.3 A</td>
</tr>
<tr>
<td>( f_n )</td>
<td>37 Hz</td>
</tr>
</tbody>
</table>
A.3 Leonard+Bauer GEL 214 interpolator

General Information
- conversion of standardized sine signals into square signals up to an interpolation coefficient of 40
- automatic offset and amplitude calibration
- storage of the adjusted values in an EEPROM

Range of application
- in combination with encoders generating sine-wave signals, such as e.g. GEL 295 KN
- interpolation of sine signals emitted by the MiniCoder GEL 244 KN
- interpolation of sine-shaped voltages with an amplitude of 1 V

Input Signals
- two sine-wave signals offset by 90° and their inverse signals
- signal level 250 mV per track = 1 V as differential signal
- reference signal and inverse reference signal

Output Signals
- two square-wave signals offset by 90° and their inverse signals
- reference pulse (option)
- output either with 5 V or 10...30 V signal level

Design
- EMC-proof metal housing, powder-lacquered black
- 15-pole D-subminiature appliance connector for input and output
- 12-pole circular connector (option)
## Technical data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage $U_s$</td>
<td>10 ... 30 V, option: 5 V</td>
</tr>
<tr>
<td>Power consumption without load</td>
<td>$\leq 2$ W</td>
</tr>
<tr>
<td>Output level</td>
<td>High level: $1.0$ V if $I = 10$ mA; $1.2$ V if $I = 30$ mA</td>
</tr>
<tr>
<td></td>
<td>Low level: $0.9$ V if $I = 10$ mA; $1.0$ V if $I = 30$ mA</td>
</tr>
<tr>
<td>V-signal, X-signal</td>
<td>High level: $1.9$ V if $I = 10$ mA; $2.8$ V if $I = 30$ mA</td>
</tr>
<tr>
<td></td>
<td>Low level: $1.2$ V if $I = 10$ mA; $2.6$ V if $I = 30$ mA</td>
</tr>
<tr>
<td>Outputs signal</td>
<td>Two square-wave signals offset by 90° and their inverse signals (option)</td>
</tr>
<tr>
<td>Outputs (T, TN)</td>
<td>TTL-, RS 422- and RS 485-compatible</td>
</tr>
<tr>
<td>Outputs (U, V, X)</td>
<td>Push pull signal</td>
</tr>
<tr>
<td>Input tracks</td>
<td>Sine-wave/cosine signals and their inverse signals</td>
</tr>
<tr>
<td>Output frequency</td>
<td>$\leq 200$ kHz</td>
</tr>
<tr>
<td></td>
<td>Input frequency by multiplier or selector switch: 1, 2, 4, 6, 8, 10, 12, 16, 20, 24, 32 or 40</td>
</tr>
<tr>
<td>Input frequency</td>
<td>0 ... 50 kHz</td>
</tr>
<tr>
<td>Short-wave precision</td>
<td>0.0085&quot;, pairing of sensor/encoder not necessary</td>
</tr>
<tr>
<td>Long-wave precision</td>
<td>Dependent on the precision of the measuring scale</td>
</tr>
<tr>
<td>Max. admissible cable length</td>
<td>25 m if the cable cross section is 0.5 mm²</td>
</tr>
<tr>
<td>Operating temperature range</td>
<td>$-40°C ... 85°C$</td>
</tr>
<tr>
<td>Protection class</td>
<td>IP 40</td>
</tr>
<tr>
<td>Directive EMC 89/336/EEC of the</td>
<td>EN 50081-1 and -2, EN 50082-1 and -2</td>
</tr>
<tr>
<td>European Union</td>
<td>Rail vehicles ENV 50121-4</td>
</tr>
<tr>
<td>Screening</td>
<td>Screen coaxial on connector housing</td>
</tr>
<tr>
<td>Insulation strength</td>
<td>500 V</td>
</tr>
<tr>
<td>Vibration-proof (EN 50155)</td>
<td>20 g</td>
</tr>
<tr>
<td>Housing made of</td>
<td>Metal</td>
</tr>
<tr>
<td>Colour</td>
<td>RAL 9005, black</td>
</tr>
<tr>
<td>Weight</td>
<td>Approx. 0.5 kg</td>
</tr>
<tr>
<td>Connectors</td>
<td>15-pole D-subminiature socket (input for the sine encoder)</td>
</tr>
<tr>
<td></td>
<td>15-pole D-subminiature connector (output for the control)</td>
</tr>
<tr>
<td></td>
<td>12-pole circular connector (option)</td>
</tr>
<tr>
<td></td>
<td>Connection to earth set screw M5</td>
</tr>
</tbody>
</table>
Dimensioned drawings

Pin layouts

**Standard:** pin layout, dimensioned drawing

<table>
<thead>
<tr>
<th>socket (input for the sine encoder)</th>
<th>connector (output for the control)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 V (GND)</td>
<td>0 V (GND)</td>
</tr>
<tr>
<td>2 -5 V DC</td>
<td>-10...30 V DC (color: +5 V DC)</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6 N (black)</td>
<td>track 1</td>
</tr>
<tr>
<td>7 U4 (track 2)</td>
<td>track 2</td>
</tr>
<tr>
<td>8 U4 (track 1)</td>
<td>N (black)</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GND - store</td>
</tr>
<tr>
<td>13 / N (black)</td>
<td>track 1</td>
</tr>
<tr>
<td>14 / U4 (track 2)</td>
<td>track 2</td>
</tr>
<tr>
<td>15 / U4 (track 1)</td>
<td>N (black)</td>
</tr>
</tbody>
</table>

See to it that the cable screening has large-surface contact with the connector housing.

*) If the cable length is >10 m, please use a cable with larger cross section, e.g. 60 m > 1.0 mm².
Selector switch Type code

Selector switch (option)

The selector switch is accessible after removal of the backplate.

Interpolation coefficient

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- switch locked

- switch open

lock J7 or J8 exchange of the sense of rotation

Type code GEL

<table>
<thead>
<tr>
<th>214</th>
<th>XX</th>
<th>XX</th>
<th>X</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>standard, 15-pole D-subminiature connector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>option, 12-pole circular connector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>interpolation coefficient 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>interpolation coefficient 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>interpolation coefficient 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>interpolation coefficient 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>interpolation coefficient 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>interpolation coefficient 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>interpolation coefficient 12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>interpolation coefficient 16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>interpolation coefficient 20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>interpolation coefficient 24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>interpolation coefficient 32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>interpolation coefficient 40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>selector switch (option)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

signal pattern

| T-  | square-wave signal 5 V |
| TN  | square-wave signal 5 V with digital reference pulse 5 V |
| V-  | square-wave signal 10 ... 30 V |
| VN  | square-wave signal 10 ... 30 V with digital reference pulse 10 ... 30 V |
| U-  | square-wave signal 5 V, U_p = 10 ... 30 V |
| UN  | square-wave signal 5 V, U_p = 10 ... 30 V with digital reference pulse 5 V |
| X-  | square-wave signal 10 ... 30 V (inverse) |
| XN  | square-wave signal 10 ... 30 V (inverse) with digital reference pulse 10 ... 30 V |
Appendix B

Derivation of the dynamical model

In this Appendix the dynamical model of the Furuta pendulum will be derived using the Lagrange's equations. In chapter 3 the set of generalized coordinates is chosen as:

\[ q = \left( \begin{array}{c} \alpha \\ \beta \end{array} \right) \]  \hspace{1cm} (B.1)

The first step in deriving the equations of motion is to determine the total kinetic energy of the system. This can be done by summing the kinetic energy of the arm and the rod. First the position of the center of mass of the pendulum rod must be expressed in the generalized coordinates. The total kinetic energy of the actuated arm is considered to consist entirely of rotational energy and thus can be expressed as a function of \( J_1 \) and \( \dot{\alpha} \) only. The position vector of the center of mass of the pendulum rod is:

\[ \ell_{cm2} = \begin{pmatrix} -l_1 \sin \alpha - l_2 \cos \alpha \sin \beta \\ \ell_1 \cos \alpha - l_2 \sin \alpha \sin \beta \\ -l_2 \cos \beta \end{pmatrix} \]  \hspace{1cm} (B.2)

The velocity, needed to compute the kinetic energy, can be found by differentiating B.2 once, by time.

\[ \dot{\ell}_{cm2} = \begin{pmatrix} -\dot{\ell}_1 \cos \alpha + \dot{\alpha}l_2 \sin \alpha \sin \beta - \dot{\beta}l_2 \cos \alpha \cos \beta \\ -\dot{\ell}_1 \sin \alpha - \dot{\alpha}l_2 \cos \alpha \sin \beta - \dot{\beta}l_2 \sin \alpha \cos \beta \\ \dot{\beta}l_2 \sin \beta \end{pmatrix} \]  \hspace{1cm} (B.3)

The total kinetic energy of the system can now be written as:

\[ T = \alpha \dot{\beta} \]  \hspace{1cm} (B.4)

When equations B.3 and B.4 are combined the total kinetic energy in terms of the generalized coordinates can be written as:

\[ T = \frac{1}{2} J_1 \dot{\alpha}^2 + \frac{1}{2} m_2 \dot{\ell}_{cm2}^T \dot{\ell}_{cm2} + \frac{1}{2} J_2 \dot{\beta}^2 \]  \hspace{1cm} (B.5)

Because there are no elastic elements present in the system, the total potential energy of the system in terms of the generalized coordinates is generated only by gravity and is given by:

\[ V = -m_2 g l_1 \cos \beta \]  \hspace{1cm} (B.6)

To be able to model the viscous damping, virtual work is applied to the system. First the work of the virtual displacement \( \delta q^T = [\delta \alpha, 0] \) is computed:

\[ \delta W = -b_1 \dot{\alpha} \delta \alpha \]  \hspace{1cm} (B.7)
And the work of the virtual displacement $\delta_q^T = [0, \delta\beta]$ is:

$$\delta W = -b_2 \delta\beta \delta\beta$$ (B.8)

With equations B.7 and B.8 and the fact that the pendulum arm is directly actuated, the column of nonconservative generalized torques becomes:

$$Q^{nc} = \begin{bmatrix} -b_1 \dot{\alpha} + T_m \\ -b_2 \beta \end{bmatrix}$$ (B.9)

In which $T_m$ equals the torque exerted by the motor on the system. The most common form of the Lagrange's equations is given by:

$$\frac{d}{dt} (T_q) - T_q + V_q = Q^{nc}$$ (B.10)

When equations B.5, B.6 and B.9 are substituted in equation B.10 the equations of motion for the Furuta pendulum are given by:

$$\begin{bmatrix} (J_1 + m_2l_2^2 + m_2l_2^2 \sin^2 \beta) \dot{\alpha} + 2m_2l_2^2 \dot{\alpha} \beta \sin \beta \cos \beta + m_2l_1l_2 \dot{\beta} \cos \beta - m_2l_1l_2 \dot{\beta}^2 \sin \beta \\ (J_2 + m_2l_2^2) \ddot{\beta} + m_2l_1l_2 \ddot{\beta} \cos \beta - m_2l_1l_2 \dot{\beta} \beta \sin \beta \\ m_2l_2^2 \dot{\alpha} \sin \beta \cos \beta - m_2l_1l_2 \dot{\beta} \sin \beta \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ m_2g \sin \beta \end{bmatrix} = \begin{bmatrix} -b_1 \dot{\alpha} + T_m \\ -b_2 \beta \end{bmatrix}$$ (B.11)

This equation can be written in a more standard form like:

$$M \ddot{q} + D \dot{q} + F = f$$ (B.12)

with:

$$M = \begin{bmatrix} J_1 + m_2l_2^2 + m_2l_2^2 \sin^2 \beta & m_2l_1l_2 \cos \beta \\ m_2l_1l_2 \cos \beta & J_2 + m_2l_2^2 \end{bmatrix}$$

$$D = \begin{bmatrix} b_1 + m_2l_2^2 \dot{\alpha} \beta \sin \beta \cos \beta & m_2l_2^2 \dot{\alpha} \beta \sin \beta \cos \beta - m_2l_1l_2 \dot{\beta} \sin \beta \\ -m_2l_2^2 \dot{\alpha} \beta \sin \beta \cos \beta & b_2 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 \\ m_2g \sin \beta \end{bmatrix}$$

$$f = \begin{bmatrix} T_m \\ 0 \end{bmatrix}$$

The behavior of the Furuta pendulum during experiments shows an obvious presence of Coulomb friction. For small input torques, exerted by the motor, the pendulum arm does not start to move, which is due to static friction present in the bearings of the arm and the gearbox. In equation B.12, the dry or Coulomb friction of the pendulum arm is not modeled yet. To introduce dry friction for the rotation $\alpha$, a standard Coulomb friction model is added to the equations of motion of the pendulum:

$$T_{fric} = C_1 \text{sign}(\dot{\alpha})$$ (B.13)

For the rotation $\beta$, the dry friction component is neglected. It is assumed that the friction in the rotation $\beta$ is completely modeled by viscous friction. Even if a small amount of Coulomb friction is present in the bearings of the pendulum rod, the dynamics will not change noticeably. The small viscous friction and Coulomb friction in the pendulum rod, are relatively unimportant for the dynamical behavior of the Furuta pendulum. Also aerodynamic drag is neglected in the equations of motion.
The input torque, $T_m$, exerted by the motor is dependent on the dynamical behavior of all the hardware between the motor and the computer. In chapter 2 we have already seen that the transfer of the current amplifier, from voltage to current, is linear in the frequency range at which the pendulum will operate. Also the acquisition systems have a more or less linear behavior in this range. The torque exerted by the motor can be modeled with a constant multiplied with output of the TUEDACs.

$$T_m = K_u u$$

With equations B.13 and B.14 the complete equations of motion for the Furuta pendulum become:

$$M \ddot{\theta} + D \dot{\theta} + F = u$$

with:

$$M = \begin{bmatrix}
J_1 + m_2 \ell_2^2 + m_2 \ell_2^2 \sin^2 \beta & m_2 \ell_1 \ell_2 \cos \beta \\
m_2 \ell_1 \ell_2 \cos \beta & J_2 + m_2 \ell_2^2
\end{bmatrix}$$

$$D = \begin{bmatrix}
b_1 + m_2 \ell_2^2 \beta \sin \beta \cos \beta & m_2 \ell_1 \ell_2 \beta \sin \beta \cos \beta - m_2 \ell_1 \ell_2 \beta \sin \beta \\
-m_2 \ell_1 \ell_2 \beta \sin \beta \cos \beta & b_2
\end{bmatrix}$$

$$F = \begin{bmatrix}
C_1 \text{sign}(\dot{\alpha}) \\
m_2 g \ell_2 \sin \beta
\end{bmatrix}$$

$$f = \begin{bmatrix}
K_u \\
0
\end{bmatrix}$$
Appendix C

The Sliding Mode Controller

In this Appendix the derivation for the used SMC is derived. Some proofs and stability analysis' are not presented in this section to keep the derivation short. They can be found in [SL91] as well as all further information about SMC and MRAC. The difference of a SMC with respect to a standard MRAC is that so-called a switching parameter $s$ is derived. This parameter defines the switching plane and is a function of the tracking error and derivatives. A representation of this switching parameter is given in figure C.1. The idea of a SMC is that $s$, provided chosen well, makes sure that whenever the system has reached a certain 'error-level' and enters the switching plane, the system can not escape it anymore. The figure shows two cases. In the left and ideal case, the system enters $s$ and does not leave it anymore. The dotted solution shows a certain band around $s$ is entered. The solution stays inside this band and may converge. Whenever the system switches to the switching plane it is said to go into 'Sliding Mode'. The most difficult part of the SMC is finding a switching parameter in combination with an adaption law that ensures global stability.

![Figure C.1: Two cases of switching to sliding mode](image-url)
C.1 Derivation of control law

The construction of a SMC starts with the model of the plant, in this case the pendulum without the pendulum rod, and the choice of a reference model. The reference model is chosen as a standard mass-spring-damper system.

\[
T_m = J_1 \dot{\alpha} + b_1 \dot{\alpha} + C_1 \text{sign}(\dot{\alpha})
\]

\[
r = \frac{1}{\omega_m^2} \dot{\alpha}_m + \frac{2\xi}{\omega_m} \dot{\alpha}_m + \alpha_m = J_m \dot{\alpha}_m + b_m \dot{\alpha}_m + k_m \alpha_m
\]

With \( r \) the reference signal. Because it is intended that the real pendulum will follow the reference signal eventually it is logical to choose the reference model of the same order as the model of the pendulum. For now it is assumed the model parameters are known. For the derivation it is easier to write equation C.1 and C.2 in standard state space format:

\[
\begin{bmatrix}
\dot{\alpha}
\end{bmatrix}, \quad \begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix}, \quad \begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix}
\]

\[
\begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix} = \begin{bmatrix}
\frac{b_m}{J_m} \alpha_m - \frac{k_m}{J_m} \text{sign}(\dot{\alpha}_m)
\end{bmatrix} + \begin{bmatrix}
0
\end{bmatrix} T_m = A(x) + B(x)T_m
\]

\[
\begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix} = \begin{bmatrix}
\frac{b_m}{J_m} \alpha_m - \frac{k_m}{J_m} \alpha_m
\end{bmatrix} + \begin{bmatrix}
0
\end{bmatrix} \alpha_m = A_m(\alpha_m) + B_m(\alpha_m)r
\]

And the error equation can be written as:

\[
\begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix} = \begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix} = \dot{x} - \dot{x}_m, \quad \dot{\alpha}_m = \dot{x} - \dot{x}_m
\]

The switching parameter will now be defined as:

\[
s = S_1 \dot{\alpha}_m + S_2 \dot{\alpha}_m
\]

To examine the stability of \( s \) the following coordinate transformation on the error equation can be implemented:

\[
\begin{bmatrix}
\eta
\end{bmatrix} = \begin{bmatrix}
\eta(x)
\end{bmatrix} = \begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix}, \quad N = \begin{bmatrix}
\delta_n
\end{bmatrix} = \begin{bmatrix}
1
0
\end{bmatrix}, \quad S = \begin{bmatrix}
\delta_s
\end{bmatrix} = \begin{bmatrix}
S_1
S_2
\end{bmatrix}
\]

This leads to:

\[
\begin{bmatrix}
\dot{\eta}
\end{bmatrix} = \begin{bmatrix}
N \dot{\alpha}_m
\end{bmatrix} = \begin{bmatrix}
N(A - \dot{x}_m) + NB_{1m}
\end{bmatrix}
\]

It is easy to see that the control law that stabilizes the second equation looks like:

\[
T_m = (SB)^{-1} [A(\dot{x}_m) - \Lambda \text{sign}(s)]
\]

The closed loop system equation becomes:

\[
\begin{bmatrix}
\dot{\alpha}_m
\end{bmatrix} = \begin{bmatrix}
N[I - B(SB)^{-1}S] (A - \dot{x}_m) - NB(SB)^{-1} \Lambda \text{sign}(s)
\end{bmatrix}
\]

After finite time \( s(t) = 0 \) and the system is said to enter the 'Sliding Mode', and the system reduces to:

\[
\dot{\alpha}_m = N[I - B(SB)^{-1}S] (A - \dot{x}_m)
\]
APPENDIX C. THE SLIDING MODE CONTROLLER

It can be shown that \( n = 0 \) is an equilibrium point of the system in sliding mode. And also that with a proper choice of \( S \) the point \( n = 0 \) can be made global asymptotically stable. This will not be discussed here. With equation C.3 and C.9 the control law, equation C.11 becomes:

\[
T_m = J_1 \left[ S_1 (\dot{\alpha}_m - \dot{\alpha}) \right] + J_1 \left[ \frac{b_m}{J_m} \dot{\alpha}_m - \frac{k_m}{J_m} \alpha_m + \frac{r}{J_m} \right] + b_1 \dot{\alpha} + C_1 \text{sign}(\dot{\alpha}) - \Lambda \text{sign}(s) \quad (C.14)
\]

And with equation C.2 this reduces to:

\[
T_m = \frac{J_1}{S_2} S_1 \dot{\alpha}_m + J_1 \dot{\alpha}_m + b_1 \dot{\alpha} + C_1 \text{sign}(\dot{\alpha}) - \Lambda \text{sign}(s) \quad (C.15)
\]

But because the model parameters are actually unknown the model parameters used in the control law like (3.15) must be replaced by their estimators. Moreover because the effect of measurement noise on the term \( \dot{\alpha} \) can be disruptive for the adaptation of the parameters and for the performance of the SMC the \( \dot{\alpha} \) terms are replaced by \( \dot{\alpha}_m \). To overcome numerical problems and avoid chattering the sign function is replaced by a saturation function. This leads to the final control law for the used SMC:

\[
T_m = -\frac{J_1}{S_2} S_1 \dot{\alpha}_m + \dot{J}_1 \dot{\alpha}_m + b_1 \dot{\alpha}_m + C_1 \text{sign}(\dot{\alpha}_m) - \Lambda \text{sat}(s) \quad (C.16)
\]

But because the introduction of the estimators the global asymptotic stability in \( s \) is not guaranteed. It can be proven that asymptotic stability can be guaranteed for:

\[
T_m = -\frac{J_1}{S_2} S_1 \dot{\alpha}_m + \dot{J}_1 \dot{\alpha}_m + b_1 \dot{\alpha}_m + C_1 \text{sign}(\dot{\alpha}_m) - (\Lambda + \epsilon) \text{sat}(s, \phi) \quad (C.17)
\]

In which \( \epsilon \) is representing the upper limit of the absolute value of the influence of model errors and disturbances. For the SMC finally used it holds that \( \epsilon < \Lambda \). Therefore \( \epsilon \) has been discarded.

The result, equation C.16 is also given as equation 4.3 in section 4.3.

C.2 Derivation of adaptation mechanism

The derivation of the adaptation mechanism for the SMC starts with the adaptation mechanism for a standard MRAC. Again the model parameters are assumed to be known. The error equation can be written as, see equation C.1 and C.2:

\[
e_\alpha = \ddot{\alpha} - \dot{\alpha}_m
\]

\[
e_\alpha = \frac{1}{J_1} (T_m - b_1 \dot{\alpha} - C_1 \text{sign}(\dot{\alpha})) - \frac{1}{J_m} (r - b_m \dot{\alpha}_m - k_m \alpha_m)
\]

\[
e_\alpha = \frac{1}{J_1} (T_m - b_1 \dot{\alpha} - C_1 \text{sign}(\dot{\alpha})) - \frac{1}{J_m} (r - b_m \dot{\alpha} - \dot{e}_\alpha - k_m (\alpha - e_\alpha)) \quad (C.18)
\]

And after rearranging:

\[
e_\alpha + \frac{b_m}{J_m} e_\alpha + \frac{k_m}{J_m} e_\alpha = \frac{1}{J_1} (T_m - b_1 \dot{\alpha} - C_1 \text{sign}(\dot{\alpha})) - \frac{1}{J_m} (r - b_m \dot{\alpha} - k_m \alpha) \quad (C.19)
\]

The left hand side of equation C.19 has to be zero to make the error equation globally asymptotically stable. Therefore the control law has to be:

\[
T_m = b_1 \dot{\alpha} + C_1 \text{sign}(\dot{\alpha}) + \frac{J_1}{J_m} (r - b_m \dot{\alpha} - k_m \alpha) \quad (C.20)
\]

But as we saw in the preceding section the model parameters in equation C.20 are not known and they have to be substituted with their estimators:

\[
T_m = \dot{b}_1 \dot{\alpha} + \dot{C}_1 \text{sign}(\dot{\alpha}) + \frac{J_1}{J_m} (r - b_m \dot{\alpha} - k_m \alpha) \quad (C.21)
\]
APPENDIX C. THE SLIDING MODE CONTROLLER

When this control law is substituted in equation C.19 the error equation becomes:

\[ J_1 \left( \dot{e}_a + \frac{b_m}{J_m} \dot{e}_a + \frac{k_m}{J_m} e_a \right) = (b_1 - b_1) \dot{\hat{\alpha}} - (C_1 - C_1) \text{sign}(\dot{\hat{\alpha}}) - \frac{J_1 - J_1}{J_m} (r - b_m \dot{\hat{\alpha}} - k_m \alpha) \]  (C.22)

The right hand side equals zero if the model parameters equal their estimators. Otherwise the stability of the error equation is not guaranteed. The goal is to find an adaptation mechanism that makes \( e_a = 0 \) an asymptotically stable equilibrium point and that assures parameter convergence. The stability of equation C.22 will be examined using the method of Lyapunov. First use a compact notation:

\[ J_1 \left( \dot{\bar{e}}_a + \frac{b_m}{J_m} \dot{\bar{e}}_a + \frac{k_m}{J_m} e_a \right) = \left( \dot{\bar{\theta}} - \theta \right)^T \nu \]  (C.23)

where:

\[ \nu^T = \left[ \bar{\alpha}_m - \frac{b_m}{J_m} \bar{\hat{\alpha}} - \frac{b_m}{J_m} e_a \ \dot{\hat{\alpha}} \ \text{sign}(\dot{\hat{\alpha}}) \right] \]

\[ \theta^T = [ J_1 \ b_1 \ C_1 ] \]

Choose as candidate Lyapunov function:

\[ V_m = J_1 \left( \dot{\bar{e}}_a + \frac{b_m}{J_m} \dot{\bar{e}}_a + \frac{k_m}{J_m} e_a \right)^2 \]  (C.24)

Then the time derivative of this Lyapunov function becomes:

\[ \dot{V}_m = -b_m \frac{J_1}{J_m} \dot{\bar{e}}_a^2 + \left( \dot{\bar{\theta}} - \theta \right)^T \nu \dot{\bar{e}}_a \]  (C.25)

The first term of equation C.25 is always negative but the second term has an undetermined sign. The use of an adaptation mechanism can force the sign of \( V_m \) by using:

\[ \dot{\hat{\theta}} = -\Gamma^{-1} \nu \dot{\bar{e}}_a \]  (C.26)

Now with \( \dot{\hat{\theta}} \) equation C.25 can be rewritten as:

\[ \dot{V}_m = -b_m \frac{J_1}{J_m} \dot{\bar{e}}_a^2 + \left( \dot{\bar{\theta}} - \theta \right)^T \Gamma \left( \dot{\bar{\theta}} - \theta \right) \]  (C.27)

With the use of \( \dot{\bar{\theta}} - \theta = e_\theta \) the second term is equal to:

\[ \frac{1}{2} \frac{\delta}{\delta t} \left( e_\theta^T \Gamma e_\theta \right) \]  (C.28)

And asymptotic stability is now assured with:

\[ V_m = \frac{1}{2} \left( J_1 \dot{\bar{e}}_a^2 + k_m \frac{J_1}{J_m} \dot{\bar{e}}_a^2 + e_\theta^T \Gamma e_\theta \right) \]  (C.29)

\[ \dot{V}_m = -b_m \frac{J_1}{J_m} \dot{\bar{e}}_a^2 \]  (C.30)

With this Lyapunov function stability is guaranteed. It is not guaranteed that \( e_a \) and \( e_\theta \) will approach zero for \( t \to \infty \). The only guarantee is that they will reach a constant value. A sufficient but not necessary condition needed to guarantee convergence to zero is that \( \nu \) has to be 'persistently exiting'. From equation C.22 it can be seen which role the reference model plays in the error dynamics. For example if the steady state error has to be decreased the factor \( \frac{b_m}{J_m} \) has to increase. This means that \( J_m \) should become smaller. This can be done by increasing the bandwidth of the reference model by increasing \( w_m \), see equation C.2. This means that the error equation becomes relatively stiff. It is obvious that if \( \Gamma^{-1} \) is also relatively large, and estimates are updated very
fast, the system may become unstable. Also the reference model has to be chosen very carefully.

Now the standard adaptation law for an MRAC controller is derived and is given in equation C.26. A disadvantage of this mechanism is that it is very sensitive to measurement noise and this will eventually lead to drift of the estimators of the model parameters. A measure that was also used in the preceding subsection is the replacement of the $\dot{\alpha}$ with $\dot{\alpha}_m$. Another measure is to remove terms with $\dot{\alpha}^2$ from the adaptation law. In this case this can be done by removing the P and D actions from the $\nu$ term. Of course when the adaptation law is altered a new stability analysis has to be done to guarantee asymptotic stability. The final adaptation law used with the inverted pendulum, in which the above measures are clearly visible, is given by:

$$\dot{\theta} = -\Gamma^{-1}W^T \hat{\theta}$$

where:

$$W = \begin{bmatrix} \ddot{\alpha}_m & \dot{\alpha}_m & \text{sign}(\dot{\alpha}_m) \end{bmatrix}$$

$$s = S_1 e_\alpha + S_2 \dot{e}_\alpha$$

This adaptation mechanism is also given as equation 4.2 in section 4.3.
C.3 SMC results

In this section all successive experiments with the SMC are presented. The values of the variables used during the successive experiments are presented in table C.1. The experimental results are presented in figures C.2 to ??.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Experiment 1</th>
<th>Experiment 2</th>
<th>Experiment 3</th>
<th>Experiment 4</th>
<th>Experiment 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi_m$</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>$\omega_m$</td>
<td>20$\pi$</td>
<td>30$\pi$</td>
<td>30$\pi$</td>
<td>40$\pi$</td>
<td>40$\pi$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$\phi$</td>
<td>0.1</td>
<td>0.5</td>
<td>0.15</td>
<td>0.20</td>
<td>0.20</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>diag(100)</td>
<td>diag(50)</td>
<td>diag(50,10,10)</td>
<td>diag(50,10,50)</td>
<td>diag(50,50,50)</td>
</tr>
<tr>
<td>$J_{1,0}$</td>
<td>45</td>
<td>35</td>
<td>35</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>$\hat{J}_{1,0}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\hat{C}_{1,0}$</td>
<td>1</td>
<td>0.85</td>
<td>0.75</td>
<td>0.73</td>
<td>0.68</td>
</tr>
<tr>
<td>max($e_n$)</td>
<td>0.002</td>
<td>0.002</td>
<td>0.0017</td>
<td>0.0015</td>
<td>0.0015</td>
</tr>
<tr>
<td>max($e_n$)</td>
<td>0.12</td>
<td>0.11</td>
<td>0.11</td>
<td>0.11</td>
<td>0.11</td>
</tr>
</tbody>
</table>

Table C.1: Variables values for the 5 experiments

Figure C.2: Parameter estimates for experiment 1
Appendix D

Simulink models

D.1 Coupling of pendulum to computer

Figure D.1: Simulink subsystem of real pendulum
D.2 Total pendulum model

Figure D.2: Simulink total Furuta pendulum model
Appendix E

Linearisation of the non-linear model

E.1 Linearisation for upright position

The linearisation of the non-linear model of the Furuta pendulum is based on Taylor series expansion of the model around the desired (equilibrium) point, $z$, and retaining only the first order terms. For a system containing $n$ states $x_1, x_2, \ldots, x_n$ the Taylor series can be expressed as:

$$y = f(x)$$
$$\bar{y} = \bar{f}(\bar{x})$$

$$y - \bar{y} = (x_1 - \bar{x}_1) \frac{\partial f}{\partial x_1} \bigg|_{x=\bar{x}} + (x_2 - \bar{x}_2) \frac{\partial f}{\partial x_2} \bigg|_{x=\bar{x}} + \ldots + (x_n - \bar{x}_n) \frac{\partial f}{\partial x_n} \bigg|_{x=\bar{x}}$$

Starting with the non-linear model, as defined in B.15, and assuming Coulomb friction compensation the equations of motion read:

$$M\ddot{q} + D\dot{q} + F = f(u) \quad (E.1)$$

$$M = \begin{bmatrix}
J_1 + m_2 l_2^2 + m_2 l_2^2 \sin^2 \beta & m_2 l_1 l_2 \cos \beta \\
 m_2 l_1 \cos \beta & J_2 + m_2 l_1^2
\end{bmatrix}$$

$$D = \begin{bmatrix}
b_1 + m_2 l_2 \dot{\alpha} \sin \beta \cos \beta & m_2 l_2 \dot{\alpha} \sin \beta \cos \beta - m_2 l_1 l_2 \dot{\beta} \sin \beta \\
- m_2 l_2 \dot{\alpha} \sin \beta \cos \beta & b_2
\end{bmatrix}$$

$$F = \begin{bmatrix}
m_2 g l_2 \sin \beta
\end{bmatrix}$$

$$f = \begin{bmatrix}
K_u \sin \beta
\end{bmatrix}$$

In fact this equation consists of two parts, one for the rotation $\alpha$ and one equation for the rotation $\beta$. The equations of motion can thus also be written as:

$$f_1(\alpha, \dot{\alpha}, \beta, \dot{\beta}) = (J_1 + m_2 l_2^2 + m_2 l_2^2 \sin^2 \beta) \dot{\alpha} + m_2 l_1 l_2 \dot{\beta} \cos \beta + b_1 \dot{\alpha} + 2m_2 l_2 l_2 \dot{\alpha} \dot{\beta} \sin \beta \cos \beta - m_2 l_1 l_2 \dot{\beta} \sin \beta - K_u u \quad (E.2)$$

$$f_2(\alpha, \dot{\alpha}, \beta, \dot{\beta}) = m_2 l_1 l_2 \dot{\alpha} \cos \beta + (J_2 + m_2 l_2^2) \dot{\beta} - m_2 l_1 l_2 \dot{\alpha}^2 \sin \beta \cos \beta + b_2 \dot{\beta} + m_2 g l_2 \dot{\beta} \sin \beta \quad (E.3)$$
APPENDIX E. LINEARISATION OF THE NON-LINEAR MODEL

Linearisation of equation E.2 around the unstable equilibrium \( x_{eq} = [0 \ 0 \ \pi \ 0]^T \) gives:

\[
\overline{f}_1(0,0,\pi,0) = (J_1 + m_2 l_1^2) \ddot{\alpha} - m_2 l_1 l_2 \ddot{\beta} - K_u u
\]

(E.4)

And for the partial derivatives follows:

\[
\left. \frac{\partial f_1}{\partial \alpha} \right|_{x=x_0} = 0, \quad \left. \frac{\partial f_1}{\partial \dot{\alpha}} \right|_{x=x_0} = b_1, \quad \left. \frac{\partial f_1}{\partial \beta} \right|_{x=x_0} = 0, \quad \left. \frac{\partial f_1}{\partial \dot{\beta}} \right|_{x=x_0} = 0
\]

So the linearized equation of motion for the rotation \( \alpha \) is given by:

\[
(J_1 + m_2 l_1^2) \ddot{\alpha} - m_2 l_1 l_2 \ddot{\beta} + b_1 \ddot{\beta} - K_u u = 0
\]

(E.5)

With \( \ddot{x} = [\ddot{\alpha} \ \ddot{\beta} \ \ddot{\theta} \ \ddot{\phi}]^T \) the state variables with respect to the equilibrium state. So \( \ddot{x} = \dot{x} - \ddot{x}_0 \).

Linearisation of equation E.3 around the unstable equilibrium \( x_{eq} = [0 \ 0 \ \pi \ 0]^T \) gives:

\[
\overline{f}_2(0,0,\pi,0) = -m_2 l_1 l_2 \ddot{\alpha} + (J_2 + m_2 l_2^2) \ddot{\beta}
\]

(E.6)

And for the partial derivatives follows:

\[
\left. \frac{\partial f_2}{\partial \alpha} \right|_{x=x_0} = 0, \quad \left. \frac{\partial f_2}{\partial \dot{\alpha}} \right|_{x=x_0} = 0, \quad \left. \frac{\partial f_2}{\partial \beta} \right|_{x=x_0} = -m_2 g l_2, \quad \left. \frac{\partial f_2}{\partial \dot{\beta}} \right|_{x=x_0} = b_2
\]

So the linearized equation of motion for the rotation \( \beta \) is given by:

\[
-m_2 l_1 l_2 \ddot{\alpha} + (J_2 + m_2 l_2^2) \ddot{\beta} - m_2 g l_2 \ddot{\beta} + b_2 \ddot{\beta} = 0
\]

(E.7)

The linearized model for the pendulum in upright position becomes:

\[
\begin{bmatrix}
J_1 + m_2 l_1^2 & -m_2 l_1 l_2 \\
-m_2 l_1 l_2 & J_2 + m_2 l_2^2
\end{bmatrix}
\begin{bmatrix}
\ddot{\alpha} \\
\ddot{\beta}
\end{bmatrix}
+
\begin{bmatrix}
b_1 & 0 \\
0 & b_2
\end{bmatrix}
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix}
+
\begin{bmatrix}
0 & 0 \\
0 & -m_2 g l_2
\end{bmatrix}
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix}
= \begin{bmatrix}
K_u \\
0
\end{bmatrix} u
\]

(E.8)

E.2 Linearisation for downward position

For the linearisation of the downward position we start again with equations E.2 and E.3. Linearisation of equation E.2 around the stable equilibrium \( x_{eq} = [0 \ 0 \ 0 \ 0]^T \) gives:

\[
\overline{f}_1(0,0,0,0) = (J_1 + m_2 l_1^2) \ddot{\alpha} + m_2 l_1 l_2 \ddot{\beta} - K_u u
\]

(E.9)

And for the partial derivatives follows:

\[
\left. \frac{\partial f_1}{\partial \alpha} \right|_{x=x_0} = 0, \quad \left. \frac{\partial f_1}{\partial \dot{\alpha}} \right|_{x=x_0} = b_1, \quad \left. \frac{\partial f_1}{\partial \beta} \right|_{x=x_0} = 0, \quad \left. \frac{\partial f_1}{\partial \dot{\beta}} \right|_{x=x_0} = 0
\]

So the linearized equation of motion for the rotation \( \alpha \) is given by:

\[
(J_1 + m_2 l_1^2) \ddot{\alpha} + m_2 l_1 l_2 \ddot{\beta} + b_1 \ddot{\beta} - K_u u = 0
\]

(E.10)

Linearisation of equation E.3 around the stable equilibrium \( x_{eq} = [0 \ 0 \ 0 \ 0]^T \) gives:

\[
\overline{f}_2(0,0,0,0) = m_2 l_1 l_2 \ddot{\alpha} + (J_2 + m_2 l_2^2) \ddot{\beta}
\]

(E.11)
APPENDIX E. LINEARISATION OF THE NON-LINEAR MODEL

And for the partial derivatives follows:

\[ \frac{\partial f_2}{\partial \alpha} \bigg|_{x=\bar{x}} = 0, \quad \frac{\partial f_2}{\partial \alpha} \bigg|_{x=\bar{x}} = 0, \quad \frac{\partial f_2}{\partial \beta} \bigg|_{x=\bar{x}} = m_2gl_2, \quad \frac{\partial f_2}{\partial \beta} \bigg|_{x=\bar{x}} = b_2 \]

So the linearized equation of motion for the rotation \( \beta \) is given by:

\[ m_2l_1l_2\ddot{\alpha} + (J_2 + m_2l_2^2)\ddot{\beta} + m_2gl_2\beta + b_2\dot{\beta} = 0 \]  
(E.12)

And the linearized model for the pendulum in downward position becomes:

\[
\begin{bmatrix}
J_1 + m_2l_1^2 & m_2l_1l_2 \\
m_2l_1l_2 & J_2 + m_2l_2^2
\end{bmatrix}
\begin{bmatrix}
\ddot{\alpha} \\
\ddot{\beta}
\end{bmatrix}
+ \begin{bmatrix}
b_1 & 0 \\
0 & b_2
\end{bmatrix}
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 \\
0 & m_2gl_2
\end{bmatrix}
\begin{bmatrix}
\ddot{\alpha} \\
\ddot{\beta}
\end{bmatrix}
= \begin{bmatrix}
K_u \\
0
\end{bmatrix} u
\]  
(E.13)
Appendix F

Input/Output Linearisation

F.1 Design of controller for $\alpha$

Relative degree and control law

To design the I/O controller for the actuated rotation the following equations of motion are used:

$$\dot{q} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \ddot{z} = \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix}, \quad \dot{z} = \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \left[ M^{-1} \left( -\frac{\partial}{\partial q} \dot{q} - K \right) \right] + \left[ 0 \right] \frac{\partial}{\partial \tau} f\left( x \right) + gu$$

Here:

$$y = \alpha = h(x)$$

The relative degree is computed using the Lie derivative:

The relative degree is computed using the Lie derivative:

$$L_q L_f h(x) = L_q h(x) = \frac{\partial h}{\partial \tau} g = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} g = 0$$

$$L_q L_f h(x) = L_q \left( \frac{\partial f}{\partial \tau} \right) = L_q \left( \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} f \right) = L_q \dot{\alpha}$$

$$= \frac{\partial \dot{\alpha}}{\partial g} \left( 0 \right) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} g$$

$$= \frac{K_u}{J_2 + m_2 l_2^2}$$

$$= \frac{K_u}{J_2 + m_2 l_2^2} \left( J_2 + m_2 l_2^2 \sin^2 \beta - m_2 l_2^2 \cos^2 \beta \right) m_2^2 + (J_1 l_2^2 + l_1^2 J_3 + l_3^2 \sin^2 \beta J_3) m_2 + J_1 J_2$$

From this computation it can be concluded that the relative degree of the system, $r$, equals 2. It can also be concluded that the expressions for the Lie derivatives of this system gain complexity very fast and the use of the Matlab symbolic toolbox is justified. To design the I/O linearizing controller the following control law is used that makes the input to output behavior of the system linear:

$$u = \frac{1}{L_q L_f h(x)} \left( -L_q h(x) + \tau \right)$$

(F.2)
The first term is already computed. The second term reads:

$$L_f^2 h(\underline{\varepsilon}) = L_f L_f h(\underline{\varepsilon}) = L_f \dot{\underline{\varepsilon}} = [0 \ 0 \ 1 \ 0] f = f_\alpha (\underline{\varepsilon})$$ (F.3)

This expression is already too complex to be written down explicitly. The control law is even more complex.

**Zero dynamics**

To compute the zero dynamics of the system with the computed control law from the preceding section, in order to examine the stability properties for the complete pendulum, the system has to be written in so-called normal form using the following coordinate transformation:

$$\Phi : \underline{\varepsilon} \mapsto \begin{pmatrix} h(\underline{\varepsilon}) \\ L_f h(\underline{\varepsilon}) \\ \eta_1 \\ \eta_2 \end{pmatrix}$$ (F.4)

The 2 new variables $h(\underline{\varepsilon})$ and $L_f h(\underline{\varepsilon})$ will be symbolized as $\xi_1$ and $\xi_2$. The new system equations then read:

$$\begin{align*}
\dot{\xi}_1 &= \xi_2 \\
\dot{\xi}_2 &= v \\
\eta_1 &= a_1 (\xi, \eta) \\
\eta_2 &= a_2 (\xi, \eta) \\
y &= \xi_1
\end{align*}$$ (F.5)

The expressions for $a_1$ and $a_2$ will now be derived, since $\dot{\xi}_1$ and $\xi_2$ equal $\alpha$ and $\dot{\alpha}$ respectively. They can be chosen arbitrarily as long as they satisfy two conditions. They have to be independent of $\eta_1$ and $\eta_2$ because otherwise the inverse coordinate transformation would not exist, and they have to satisfy the following property:

$$\frac{\partial \eta_i}{\partial \xi} g(\underline{\varepsilon}) = 0$$ (F.6)

For a proof see [Sac96]. It's obvious that $\beta$ can be chosen as $\eta_1$, it is independent of $\xi_1$ and $\xi_2$ and satisfies equation F.6. The second function $a_2$ is more difficult to derive. The only available direct coordinate is $\dot{\beta}$. Although it is independent of all other coordinates it doesn't satisfy the second property. To derive $\eta_2$, the column $g$ is written as:

$$g = \begin{bmatrix} 0 \\ M^{-1} f \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\sigma_{q}}{k(\varepsilon)} \\ \frac{\sigma_{l}}{k(\varepsilon)} \cos \beta \\ \frac{\sigma_{l}}{k(\varepsilon)} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{P_x}{k(\varepsilon)} \\ \frac{P_x}{k(\varepsilon)} \cos \beta \\ \frac{P_x}{k(\varepsilon)} \end{bmatrix}$$

with $k(\varepsilon)$ some function of $\varepsilon$ but for both terms the same. The if $\eta_2$ is chosen as:

$$\eta_2 = P_2 \dot{\alpha} + P_1 \beta$$ (F.7)

it automatically satisfies property F.6. And this expression is also independent of all other coordinates except for $\beta = \pm \frac{\pi}{2}$. The coordinate transformation thus becomes:

$$\begin{bmatrix} \xi_1 \\ \xi_2 \\ \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} \alpha \\ \dot{\alpha} \\ \beta \\ P_2 \dot{\alpha} + P_1 \beta \end{bmatrix}$$ (F.8)
And its inverse:

\[
\begin{bmatrix}
\alpha \\
\beta \\
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} =
\begin{bmatrix}
\xi_1 \\
\eta_1 \\
\xi_2 \\
-\frac{1}{P_1} \xi_2 + \frac{1}{P_1} \eta_2
\end{bmatrix}
\] (F.9)

The system equations (F.1) in normal form than read:

\[
\begin{align*}
\dot{\xi}_1 &= \xi_2 \\
\dot{\xi}_2 &= v \\
\dot{\eta}_1 &= -\frac{P_2}{P_1} \xi_2 + \frac{1}{P_1} \eta_2 \\
\dot{\eta}_2 &= -\left(\frac{1}{J_2 + \frac{1}{2} m_2} + \frac{\gamma_2}{J_2 + \frac{1}{2} m_2}\right) l_1 l_2 m_2 \xi_2 \sin \eta_1 + \frac{\gamma_2}{J_2 + \frac{1}{2} m_2} l_1 l_2 m_2 \xi_2 \sin \eta_1 \cos \eta_1 \\
&= m_g l_2 \sin \eta_1 - b_2 \left(-\frac{l_1 l_2 m_2 \xi_2 \sin \eta_1 - \eta_2}{J_2 + \frac{1}{2} m_2}\right)
\end{align*}
\] (F.10) (F.11)

The zero dynamics are defined as the dynamics that remain if the controlled state has reached a constant setpoint. So the zero dynamics are the dynamics for \(\xi_1 = \xi_2 = v = 0\). For this system they become:

\[
\begin{align*}
\dot{\xi}_1 &= 0 \\
\dot{\xi}_2 &= 0 \\
\dot{\eta}_1 &= \frac{\eta_2}{P_1} \\
\dot{\eta}_2 &= -m_g l_2 \sin \eta_1 - \frac{b_2 \eta_2}{J_2 + \frac{1}{2} m_2}
\end{align*}
\] (F.12)

When these zero dynamics are transformed back into the original coordinates it follows that:

\[
\begin{align*}
\dot{a} &= g \\
\dot{\alpha} &= 0 \\
\dot{b} &= b \\
\dot{\beta} &= -m_g l_2 \sin \beta - b_2 \dot{\beta}
\end{align*}
\] (F.13)

This result is also presented in equation 5.27 in section 5.2.3.
F.2 Design of controller for $\beta$

Relative degree and control law

To design the I/O controller for the free rotation the same equations of motion as for the actuated rotation are used except for the output equation:

\[
\begin{bmatrix}
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} = \begin{bmatrix}
\dot{q} \\
\dot{\dot{q}}
\end{bmatrix} = \begin{bmatrix}
M^{-1}
\begin{bmatrix}
\dot{q} \\
\dot{\dot{q}}
\end{bmatrix} - K
+ \begin{bmatrix}
0 \\
M^{-1} f
\end{bmatrix} \\
&= f(x) + gu \\
y = \beta = h(x)
\end{bmatrix}
\]

with:

\[
M = \begin{bmatrix}
J_1 + m_2l_2^2 + m_2l_2^2 \sin^2 \beta & m_2l_1l_2 \cos \beta \\
m_2l_1l_2 \cos \beta & J_2 + m_2l_2^2
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
b_1 + m_2l_2^2 \beta \sin \beta \cos \beta & m_2l_1l_2 \beta \cos \beta - m_2l_1l_2 \beta \sin \beta \\
-m_2l_1 \beta \sin \beta \cos \beta & b_2
\end{bmatrix}
\]

\[
F = \begin{bmatrix}
0 \\
m_2gl_2 \sin \beta
\end{bmatrix}
\]

\[
f = \begin{bmatrix}
K_u \\
0
\end{bmatrix}
\]

To compute the relative degree of this system again the Lie derivative will be used:

\[
L_{g}\frac{\partial h}{\partial z}(x) = L_g h(x) = \frac{\partial h}{\partial z} g = [0 \ 1 \ 0 \ 0] g = 0
\]

\[
L_{g} L_{f} h(x) = L_g \left( \frac{\partial f}{\partial x} \right) = L_g \left( [0 \ 0 \ 0 \ 1] f \right) = L_g \beta
\]

\[
= \frac{\partial \beta}{\partial z} g = [0 \ 0 \ 0 \ 1] g
\]

\[
= \frac{K_u \ (-m_2l_1l_2 \cos \beta)}{\left( l_2^2 + l_2^2 \sin^2 \beta - l_2^2 \cos^2 \beta m_2^2 \right. + J_1 l_2^2 + J_2 l_2^2 + l_2^2 \sin^2 \beta)^2}
\]

From this computation it can be concluded that the relative degree of this system, $r$, also equals 2. It can also be concluded that for $\beta = \pm \frac{\pi}{2}$ the relative degree is not defined. This is logical because the system is uncontrollable in the case the pendulum rod is in horizontal position. To design the I/O linearizing controller the control law will be used that makes the input to output behavior of the system linear:

\[
u = \frac{1}{L_g L_{f} h(x)} \left( -L_{f}^2 h(x) + v \right)
\]

The second term reads:

\[
L_{f}^2 h(x) = L_{f} L_{f} h(x) = L_{f} \beta = [0 \ 0 \ 0 \ 1] f = f_\beta(x)
\]

This expression is, like the expression for the I/O controller for $\alpha$, too complex to be written down explicitly. Another conclusion is that, for the positions of the pendulum rod for which the relative degree isn't defined, the output of the control law is infinitely large.

Zero dynamics

To compute the zero dynamics of the system with the computed control law from the preceding section, in order to examine the stability properties for the complete pendulum, the system has
to be written in so-called normal form using the following coordinate transformation:

$$\Phi : \mathbf{z} \rightarrow \begin{pmatrix} h(\mathbf{z}) \\ L_f h(\mathbf{z}) \\ \eta_1 \\ \eta_2 \end{pmatrix} \tag{F.17}$$

The 2 new variables $h(\mathbf{z})$ and $L_f h(\mathbf{z})$ will be symbolized as $\xi_1$ and $\xi_2$. The new system equations then read:

$$\dot{\xi}_1 = \xi_2$$
$$\dot{\xi}_2 = v$$
$$\dot{\eta}_1 = a_1 (\xi, \eta)$$
$$\dot{\eta}_2 = a_2 (\xi, \eta)$$
$$y = \xi_1 \tag{F.18}$$

The expressions for $a_1$ and $a_2$ will now be derived, since $\xi_1$ and $\xi_2$ equal $\beta$ and $\dot{\beta}$ respectively. The can be chosen arbitrarily as long as they satisfy 2 conditions. They have to be independent of $\xi_1$ and $\xi_2$ because otherwise the inverse coordinate transformation would not exist, and they have to satisfy the following property:

$$\frac{\partial \eta_1}{\partial \xi} g(\mathbf{z}) = 0 \tag{F.19}$$

For a proof see [Sas99]. It’s obvious that $\alpha$ can be chosen as $\eta_1$, since it is independent of $\xi_1$ and $\xi_2$ and satisfies equation F.19. The second function $\alpha_2$ is more difficult to derive. The only available direct coordinate is $\alpha$. Although it is independent of all other coordinates it doesn’t satisfy the second property. To derive $\eta_2$ the column $g$ is written as:

$$g = \begin{bmatrix} 0 \\ M^{-1f} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\partial F}{\partial z} \\ \frac{\partial F}{\partial \alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\partial F}{\partial \xi} \\ \frac{\partial F}{\partial \eta} \end{bmatrix}$$

with $k(z)$ some function of $\mathbf{z}$ but for both terms the same. The if $\eta_2$ is chosen as:

$$\eta_2 = P_2 \alpha + P_1 \dot{\beta} \tag{F.20}$$

it automatically satisfies property F.19. And this expression is also independent of all other coordinates except for $\beta = \pm \frac{\pi}{2}$. The coordinate transformation thus becomes:

$$\begin{bmatrix} \xi_1 \\ \xi_2 \\ \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} \beta \\ \dot{\beta} \\ \alpha \\ P_2 \alpha + P_1 \dot{\beta} \end{bmatrix} \tag{F.21}$$

And its inverse:

$$\begin{bmatrix} \alpha \\ \beta \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} \eta_1 \\ \xi_1 \\ -\frac{P_2 \xi_2 + \frac{1}{P_1} \eta_2}{\xi_2} \end{bmatrix} \tag{F.22}$$
APPENDIX F. INPUT/OUTPUT LINEARISATION

The system equations (F.14) in normal form then read:

\[
\begin{align*}
\dot{\xi}_1 &= \xi_2 \\
\dot{\xi}_2 &= v \\
\dot{\eta}_1 &= -\frac{p_1}{p_2} \xi_2 + \frac{1}{p_2^2} \eta_2 \\
\dot{\eta}_2 &= -\xi_2 l_1 l_2 m_2 \sin \xi_1 \left( -\frac{(J_2 + I_2^2 m_2) \xi_2}{L_1 l_2 m_2 \cos \xi_2} + \frac{\eta_2}{L_1 l_2 m_2 \cos \xi_2} \right) \\
&\quad + I_2^2 m_2 \left( -\frac{\xi_2 (J_2 + I_2^2 m_2) - \eta_2}{L_1 l_2 m_2 \cos \xi_2} \right)^2 \sin \xi_1 \cos \xi_1 - m_2 g l_2 \sin \xi_1 - b_2 \xi_2
\end{align*}
\]  

(F.23)

The zero dynamics are defined as the dynamics that remain if the controlled state has reached a constant setpoint. So the zero dynamics are the dynamics for \( \xi_1 = \xi_2 = v = 0 \). For this system they become:

\[
\begin{align*}
\dot{\xi}_1 &= 0 \\
\dot{\xi}_2 &= 0 \\
\dot{\eta}_1 &= \frac{\eta_2}{p_2} \\
\dot{\eta}_2 &= 0
\end{align*}
\]  

(F.24)

When these zero dynamics are transformed back into the original coordinates it follows that:

\[
\begin{align*}
\dot{\alpha} &= \alpha \\
\dot{\alpha} &= 0 \\
\dot{\beta} &= 0 \\
\dot{\beta} &= 0
\end{align*}
\]  

(F.25)

This result is also presented in equation 5.32 in section 5.2.3.
Appendix G

Simulation and experimental results

In this Appendix the results of the simulations and experiments that are not treated in chapter 5 are discussed. First the results of the combination of the energy controller and the I/O controller for \( \alpha \) are presented. Then the simulations and experiments with the I/O linearizing controller for \( \beta \) and the linearized controller for \( \beta = 0 \) are discussed separately.

G.1 Energy controller + I/O linearizing controller \( \alpha \)

Implementation

In section 5.2.1 we have already seen that the energy based controller outputs an acceleration for the actuated rotation, \( \alpha \). In order to achieve this acceleration the I/O linearizing controller is used. In fact these two controllers are combined to form the 'Swing-up controller'. The energy controller is build with standard Simulink blocks. This is difficult for the I/O controller because of its complex expression. This controller is implemented by programming it in a C-file S-function, in order to make it available for the real-experiments. Because the I/O controller needs, besides the acceleration of the setpoint, an angular velocity and an angle, the signal from the energy controller is integrated twice. These three signals, together with the actual state \([\alpha \ \beta \ \dot{\alpha} \ \dot{\beta}]^T\) form the four inputs of the I/O controller. The output of this combination is a motor torque that achieves the desired acceleration of the pendulum arm.

Simulations and experiments

The PD controller in the control law for the I/O linearizing controller for \( \alpha \), see equation 5.26, is chosen to have a bandwidth of 3.33 Hz. The results using this Swing-up controller are visualized in figure G.1 and G.2 for two different values of \( n \), a measure for the energy increase per swing. From these figures it can be concluded that the Swing-up controller works perfectly and can swing up the pendulum rod near the upward position. It can also be seen that the Swing-up controller itself is not able to stabilize the pendulum. For \( n = 0.1 \) the pendulum rod reaches the upward position after 12 swings. Because the rod falls down and the energy controller keeps adding energy to the system the pendulum rod passes the upward position. For \( n = 1.4 \) the energy controller swings up the pendulum with only one swing and almost stabilizes the upward position. The dependence of the controller on \( n \) is clearly visible.
Figure G.1: Results for Swing-up controller, $n = 0.1$

Figure G.2: Results for Swing-up controller, $n = 1.4$
For the experiments the same initial conditions and parameter values are used as during the simulations, so a controller with a bandwidth of 3.33 Hz is used in the control law for the I/O controller. The most remarkable difference of the result with respect to the simulations is that the Swing up controller needs more swings, 16 in total, to bring the pendulum rod near the upright position. This can for instance be explained by the fact that the real pendulum suffers from drag that slows the pendulum rod down. This drag is not included in the pendulum model. So the achieved acceleration of the arm does not add as much energy to the pendulum rod as expected. Moreover the direction and angle dependency of the Coulomb friction in combination with its simple compensation can cause the pendulum arm to slow down. But this can be corrected by increasing the P and D gains in the I/O controller, to improve the tracking of the pendulum arm. Of course this increase is limited by the resonance frequencies and phase characteristics of the system. Especially the bandwidth of the open loop system is limited by the phase delay for the high frequency range. And increasing the derivative action can lead to instability due to the amplification of measurement noise. So increasing the P and D action in the controller always has be done wisely. Furthermore from the experimental results it can be concluded that the Swing-up controller itself can not stabilize the pendulum rod in upright position. By switching to a linearized controller, when the pendulum rod is close (enough) to the upright position, stabilization can be achieved.

G.2 I/O linearizing controller \( \beta \)

Implementation

The I/O linearizing controller for the free rotation, \( \beta \), should be able to let the pendulum rod track any setpoint, given this setpoint does not saturate the actuator. The maximum torque output is limited by the TUEDACs. To bring the pendulum rod in upward position a setpoint is created that takes the angle from 0 to \( \pm \pi \). The pendulum can reach the upward position in a clockwise or counterclockwise manner. The will always setpoint cross \( \pm \pi/3 \), the point for which the system is uncontrollable. The controller will generate an infinitely high output if the pendulum rod is in this point and the actuator will therefore always saturate. If the setpoint of the pendulum rod passes this uncontrollable point too slow the setpoint can not be tracked anymore. If the setpoint is fast enough though, the pendulum will just swing past the horizontal position and the controller is able to bring it in upright configuration without saturating. The generated setpoint, its velocity and acceleration profile, for the counterclockwise direction, are given in figure G.3. For the clockwise Swing-up mode the same setpoint is used except for a negative sign.

The only disadvantage of the I/O controller is that the other degree of freedom, \( \alpha \), can not be controlled and can take any value. The I/O linearizing controller for \( \beta \) is programmed in a C-file S-function because of its complexity and to make it available during the real-time experiments. The controller can be used to bring the pendulum in upright position but it can also be used to bring it from the upward position back to the downward configuration. Moreover it can be used follow an arbitrary setpoint superimposed on the up or downward position. During all simulations and experiments a third-order setpoint generator is used. This generator has the advantage of creating a setpoint with zero initial velocity and acceleration. It also provides the velocity and acceleration profiles needed by the controller. Together with the actual state of the pendulum, the controller gets four signals. The control law in the I/O controller, needed to stabilize the total system, is the same for all simulations and experiments and has a bandwidth of 17.5 Hz. Also in this case the controller parameters are limited by the characteristics of the system and stability must be guaranteed. The controller is more aggressive than the one used for the I/O controller for \( \alpha \). A explanation is that the pendulum rod itself suffers more from unmodeled dynamics like aerodynamic drag and some friction, than the pendulum arm. The pendulum arm is less sensitive to this unmodeled dynamics than the rod because its relative high inertia. So a stiffer and more aggressive controller is needed to make the pendulum rod track the setpoint accurately.
Simulations and experiments

The results for the counterclockwise Swing-up mode, using I/O linearisation, are presented in figure G.4. From these results it can be learned that the intended strategy works quite well. The setpoint passes the uncontrollable horizontal position fast enough to keep the pendulum rod track the setpoint. The tracking error is a little bit larger during the experiment than for the simulation but this can again be explained by some unmodeled dynamics and the non-perfect Coulomb friction compensation. The error signal looks like the acceleration profile and, by looking at the sign of this signal, this may be caused by overcompensating the Coulomb friction. From section 5.2.3 we know that the final speed of the pendulum arm will be constant whenever $\beta$ has reached a constant setpoint. For the simulation shown in figure G.4 the final constant speed of the pendulum arm is $\alpha_\infty = -18.4 \text{ rad/s}$. During the experiments though, the angular velocity of the pendulum arm is not constant but keeps decreasing, caused again by reasons mentioned above.

For the other modes, like shaking in upward and downward pendulum positions, similar results have been found. Although the setpoint profile is generated in a slightly different way than for the discussed swing-up procedure. A regular sine wave is multiplied with a 'ramp-like' setpoint generated by the third order setpoint generator. This signal starts with zero initial position and velocity and ramps up to 1. At the end of the shake mode the signal ramps down to zero again. This is done to make the transition to other pendulum modes smooth. The setpoint for the shake mode is presented in figure G.5.
APPENDIX G. SIMULATION AND EXPERIMENTAL RESULTS

Figure G.4: Results of swing-up by I/O controller

Figure G.5: Generated setpoint for the 'Shake' mode
Appendix G. Simulation and Experimental Results

500

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{acceleration_profiles.png}
\caption{Acceleration profiles for two swing modes}
\end{figure}

Only one last special case is presented in this section, the 'Swing-down by I/O linearisation' mode. The pendulum rod now has to track a setpoint that brings the pendulum rod from upward to downward position. Once the pendulum rod is past the horizontal position the pendulum arm has to accelerate causing the pendulum rod to decelerate and to keep track of the setpoint. This was also the case for the swing-up procedure. For the Swing-up controller the gravity helps the controller to slow the pendulum down. For the Swing-down mode the gravity works against the controller. The controller has to put extra effort to keep the pendulum rod on its track. This causes the final speed of the pendulum arm to be higher as it is for the Swing-up mode. Gravity has the most effect for pendulum positions around the horizontal position. In figure G.6 both the acceleration profiles for the Swing-up and Swing-down mode of the pendulum arm during simulations are presented. The point where the profile changes sign rapidly corresponds with the horizontal position of the pendulum rod. It is obvious that the difference between the profiles, causing the higher final velocity, is mainly concentrated around this position. A consequence of this higher velocity is that, when switching to other controller modes, the final pendulum arm velocity is too high. This initial condition for the successive mode causes the system to become unstable. To overcome this problem another setpoint is created. Whenever the Swing-down mode is performed in one stroke the problem mentioned above exists. A solution is a profile that forces the acceleration to change sign, in order to decrease the final velocity. This can be done by superimposing little swing-up's on the Swing-down profile. Every time such a swing-up arises in the setpoint the acceleration is forced to change sign. The generated setpoint used for simulations and experiments is given in figure G.7, which shows that 2 swing-ups are added. A disadvantage of this new setpoint is that it takes approximately 4 times as long to get the pendulum rod down. But the final velocity, $\omega_\infty$, of the pendulum arm has decreased from approximately 23 rad/s to 7 rad/s enabling successive modes to be executed. Figure G.8 shows the results of the simulation and experiment for this setpoint profile in counterclockwise direction.
Figure G.7: Generated setpoint for the swing-down mode by I/O control

Figure G.8: Results of swing-down by I/O controller
G.3 Linearized controller $\beta = 0$

Implementation

In this section the results for the linearized controller around the stable equilibrium point $\beta = 0$ will be discussed. This controller is modeled with the use of standard Simulink blocks, like its counterpart for the unstable pendulum position. Because now there is not a energy based controller to bring the pendulum down, first the free-fall case will be discussed. This means that the pendulum will fall down uncontrolled from the upright position and the linearized controller will try to stabilize it. The combination with the I/O controller for $\beta$ will be discussed in the next section. Again some switching algorithm has to be implemented, and its implementation is given in chapter 6.

Simulations and experiments

The region of attraction of the pendulum rod with linearized controller for the downward position plays an even more important role than it does for the upright configuration. After some simulations it can be concluded that the region of attraction for the non-linear model is approximately zero if both degrees of freedom have to be stabilized. Because the gravity now helps to accelerate the pendulum rod in its free fall, the controller has to put a lot of effort in decelerating the pendulum rod. This effort is of such magnitude that the stabilization for $\dot{\dot{\beta}}$ can not be realized at the same time. Although it seems strange, because the downward position is naturally stable, the fast stabilization of $\beta$ can not go hand in hand with stabilization for $\alpha$. The following considerations are made to be able to choose the closed loop poles of the system $\dot{\vec{x}} = (A_{down} - B_{down}K)\vec{x}$:

- The poles for $\alpha$ and $\dot{\alpha}$ have to be relative small, or even zero, to be able to stabilize the pendulum rod.
- To include damping, needed to decelerate the pendulum rod, two complex conjugate poles have to be chosen for $\beta$ and $\dot{\beta}$.
With this in mind and again the limitations with respect to the system characteristics as phase delay and resonances, mentioned in section 5.2.2, the closed loop poles for the linearized controller are chosen as \( P_{\text{down}} = [0 \ 0 \ -60 + 6i \ -60 - 6i] \). With these closed loop dynamics, and by trial and error, the best results are found if the switch from Free-fall mode to Stabilizing mode occurs at \( \pm 65^\circ \) relative to the downward position \( \beta = 0 \). To avoid chattering the switch from stabilizing to free fall mode is set at \( \pm 70^\circ \), but this would normally never happen. The results for this Free-fall mode in simulation and during the experiment are given in figure G.9. Within 1.5 seconds the pendulum rod can be stabilized with free-fall and the linearized controller. With the chosen poles the actuated rotation itself is not controlled. The pendulum arm has a constant final angular velocity, as it does in the I/O linearizing case. This final velocity equals 4.2 rad/s and 1.2 rad/s for the simulation and experiment respectively. During the simulations and experiments with the Free-fall mode, the initial pendulum arm velocity is nearly zero.
Appendix H

Switching mechanism code

H.1 Switching_mechanism.c

#define S_FUNCTION_NAME Switching_algorithm
#define S_FUNCTION_LEVEL 2

#include "simstruct.h"
#define NSTATES 0
#define NINPUTS 4
#define NOUTPUTS 13
#define NPARAMS 4

#define SWITCH_SW_TO_BAL *mxGetPr(ssGetSFcnParam(S,0))
#define SWITCH_BAL_TO_SW *mxGetPr(ssGetSFcnParam(S,1))
#define SWITCH_SWD_TO_BAL *mxGetPr(ssGetSFcnParam(S,2))
#define SWITCH_BAL_TO_SWD *mxGetPr(ssGetSFcnParam(S,3))

#include <math.h>
#include "sequence.h"

static void NextMode(SimStruct *S);

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S,NPARAMS);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
    {
        return; /* Parameter mismatch will be reported by Simulink */
    }
    ssSetNumContStates(S,NSTATES);
    ssSetNumDiscStates(S,0);
    if (!ssSetNumInputPorts(S,NINPUTS)) return;
    ssSetInputPortWidth(S,0,4);
    ssSetInputPortDirectFeedThrough(S,0,1);
    ssSetInputPortWidth(S,1,1);
    ssSetInputPortDirectFeedThrough(S,1,1);
    ssSetInputPortWidth(S,2,1);
    ssSetInputPortDirectFeedThrough(S,2,1);
    ssSetInputPortWidth(S,3,1);
    ssSetInputPortDirectFeedThrough(S,3,1);

    /* Initialise parameters for switching mechanism */
    int mode_number = -1;
    int init_mode = 1;
    int init_balance = 1;
    real_T t_start = 0;
    real_T beta_buf = 0;
}
APPENDIX H. SWITCHING MECHANISM CODE

if (!ssSetNumOutputPorts(S, NOUTPUTS)) return;
ssfSetOutputPortWidth(S, 0, 2);
ssfSetOutputPortWidth(S, 1, 2);
ssfSetOutputPortWidth(S, 2, 2);
ssfSetOutputPortWidth(S, 3, 2);
ssfSetOutputPortWidth(S, 4, 2);
ssfSetOutputPortWidth(S, 5, 2);
ssfSetOutputPortWidth(S, 6, 2);
ssfSetOutputPortWidth(S, 7, 2);
ssfSetOutputPortWidth(S, 8, 2);
ssfSetOutputPortWidth(S, 9, 2);
ssfSetOutputPortWidth(S, 10, 1);
ssfSetOutputPortWidth(S, 11, 1);
ssfSetOutputPortWidth(S, 12, 1);
ssfSetNumSampleTimes(S, 1);

/* Take care when specifying exception free code - see sfunmpl_doc.c */
ssfSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE | SS_OPTION_USE_TLC_WITH_ACCELERATOR);

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

static void mdlOutputs(SimStruct *S, int _T tid)
{
    // Initialize parameters, inputs and outputs
    // Adamant-free code

    InputRealPtrType uPtrs = ssGetInputPortRealSignalPtrs(S, 0);
    real_T *t = ssGetTPtr(S);
    real_T *motor_torque = ssGetOutputPortRealSignal(S, 0);
    real_T *swing_up_enabled = ssGetOutputPortRealSignal(S, 1);
    real_T *swing_up_speed = ssGetOutputPortRealSignal(S, 2);
    real_T *swing_up_c_enabled = ssGetOutputPortRealSignal(S, 3);
    real_T *swing_down_c_enabled = ssGetOutputPortRealSignal(S, 4);
    real_T *shake_fast_enabled = ssGetOutputPortRealSignal(S, 5);
    real_T *shake_slow_enabled = ssGetOutputPortRealSignal(S, 6);
    real_T *swing_360_c_enabled = ssGetOutputPortRealSignal(S, 7);
    real_T *swing_360_cc_enabled = ssGetOutputPortRealSignal(S, 8);
    real_T *balance_switch = ssGetOutputPortRealSignal(S, 9);
    real_T *alpha = **uPtrs++;
    real_T *alpha_dot = **uPtrs++;
    real_T *beta = **uPtrs++;
    real_T *beta_dot = **uPtrs++;
    real_T *balance_cont = **uPtrs++;
    real_T *10_alpha_cont = **uPtrs++;
    real_T *10_beta_cont = **uPtrs++;

    extern int mode_number;
    extern int init_mode;
    extern int init_balance;
extern real_T t_start;
extern real_T beta_buf;
extern struct sequence;

// ******************************************************************************
// | Start first mode |
// ******************************************************************************
if (mode_number==1){
    NextMode(S);
}

// ******************************************************************************
// | Energy based controllers |
// ******************************************************************************
else if (sequence[mode_number].name == "Swing up by energy"){
    if (fabs(beta) <= 1e-2 && fabs(beta_dot) <= 1e-1){
        *motor_torque = 70;
    } else if ((t - t_start) > sequence[mode_number].options[0]){*
        *swing_up_enabled = 0;
        NextMode(S);
    } else {
        *motor_torque = 70_alpha_cont;
        *swing_up_enabled = 1;
        *swing_up_speed = sequence[mode_number].options[1];
    }
}

else if (sequence[mode_number].name == "Swing up by energy and balance with linearized controller"){
    if (fabs(beta) <= 1e-2 && fabs(beta_dot) <= 1e-1){
        *motor_torque = 70;
    } else if (cos(beta) <= cos(SWITCH_SWU_TO_BAL)){
        *motor_torque = balance_cont;
        *balance_switch = 1;
        *swing_up_enabled = 0;
        if ((t - t_start) > sequence[mode_number].options[0]){
            NextMode(S);
        }
    } else if (cos(beta) >= cos(SWITCH_BAL_TO_SWU)){
        *motor_torque = 70_alpha_cont;
        *swing_up_enabled = 1;
        *swing_up_speed = sequence[mode_number].options[1];
    }
}

// ******************************************************************************
// | Free fall and linearized controller |
// ******************************************************************************
else if (sequence[mode_number].name == "Free fall and balance down with linearized controller"){
    if ((t - t_start) > sequence[mode_number].options[0]){*
        NextMode(S);
    } else if (cos(beta) >= cos(SWITCH_SWD_TO_BAL)){
        *motor_torque = balance_cont;
APPENDIX H. SWITCHING MECHANISM CODE

```plaintext
balance_switch = 0;
}
else if (cos(beta) <= cos(SWITCH_BAL_TO_SWD)){
  motor_torque = 0;
}
}

//******************************************************************************
// | Swing up I/O controllers (with and without linearized controller) |
//******************************************************************************
else if (sequence[node_number].name == "Swing up by IO cc-wise"){  
  if ((t - t_start) > sequence[node_number].options[0]){  
    NextMode(S);
    swing_up_cc_enabled = 0;
  } else {  
    motor_torque = IO_beta_cont;
    swing_up_cc_enabled = 1;
  }  
}
else if (sequence[node_number].name == "Swing up by IO cc-wise"){  
  if ((t - t_start) > sequence[node_number].options[0]){  
    NextMode(S);
    swing_up_cc_enabled = 0;
  } else {  
    motor_torque = IO_beta_cont;
    swing_up_cc_enabled = 1;
  }  
}
else if (sequence[node_number].name == "Swing up by IO cc-wise and balance with linearized controller"){  
  if ((t - t_start) > sequence[node_number].options[0]){  
    NextMode(S);
    swing_up_cc_enabled = 0;
  } else if (cos(beta) <= cos(SWITCH_SWU_TO_BAL)){  
    motor_torque = balance_cont;
    balance_switch = 1;
    swing_up_cc_enabled = 0;
  } else if (cos(beta) >= cos(SWITCH_BAL_TO_SWU)){  
    motor_torque = IO_beta_cont;
    swing_up_cc_enabled = 1;
  }  
}
else if (sequence[node_number].name == "Swing up by IO cc-wise and balance with linearized controller"){  
  if ((t - t_start) > sequence[node_number].options[0]){  
    NextMode(S);
    swing_up_cc_enabled = 0;
  } else if (cos(beta) <= cos(SWITCH_SWU_TO_BAL)){  
    motor_torque = balance_cont;
    balance_switch = 1;
    swing_up_cc_enabled = 0;
  } else if (cos(beta) >= cos(SWITCH_BAL_TO_SWU)){  
    motor_torque = IO_beta_cont;
    swing_up_cc_enabled = 1;
  }  
}
```
else if (sequence[node_number].name == "Swing down by cc-wise"){
    if (((t - t_start) > sequence[node_number].options[0]){ /*
      NextMode(S);
      *swing_down_cc_enabled = 0;
    } else {
      *motor_torque = IO_beta_cont;
      *swing_down_cc_enabled = 1;
    }
  }

else if (sequence[node_number].name == "Swing down by cc-wise"){
    if (((t - t_start) > sequence[node_number].options[0]){ /*
      NextMode(S);
      *swing_down_cc_enabled = 0;
    } else {
      *motor_torque = IO_beta_cont;
      *swing_down_cc_enabled = 1;
    }
  }

else if (sequence[node_number].name == "Swing down by IO cc-wise and balance with linearized controller"){
    if (((t - t_start) > sequence[node_number].options[0]){ /*
      NextMode(S);
      *swing_down_cc_enabled = 0;
    } else if (((t - t_start) > 1){ /*
      *motor_torque = balance_cont;
      *balance_switch = 0;
      *swing_down_cc_enabled = 0;
    } else {
      *motor_torque = IO_beta_cont;
      *swing_down_cc_enabled = 1;
    }
  }

else if (sequence[node_number].name == "Swing down by IO cc-wise and balance with linearized controller"){
    if (((t - t_start) > sequence[node_number].options[0]){ /*
      NextMode(S);
      *swing_down_cc_enabled = 0;
    } else if (((t - t_start) > 1){ /*
      *motor_torque = balance_cont;
      *balance_switch = 0;
      *swing_down_cc_enabled = 0;
    } else {
      *motor_torque = IO_beta_cont;
      *swing_down_cc_enabled = 1;
    }
  }
}

else if (sequence[node_number].name == "Swing down by IO cc-wise and balance with linearized controller"){
    if (((t - t_start) > sequence[node_number].options[0]){ /*
      NextMode(S);
      *swing_down_cc_enabled = 0;
    } else if (((t - t_start) > 1){ /*
      *motor_torque = balance_cont;
      *balance_switch = 0;
      *swing_down_cc_enabled = 0;
    } else {
      *motor_torque = IO_beta_cont;
      *swing_down_cc_enabled = 1;
    }
  }

// ***************************************************************************
// | Swing 360 I/O controllers  |
// ***************************************************************************
else if (sequence[mode_number].name == "Swing 360 degrees by 10 cc-wise"){
    if ((t - t_start) > sequence[mode_number].options[0]){
        NextMode(S);
        *swing_360_cc_enabled = 0;
    } else {
        *motor_torque = 10.0 * beta_cont;
        *swing_360_cc_enabled = 1;
    }
}

else if (sequence[mode_number].name == "Swing 360 degrees by 10 c-wise"){
    if ((t - t_start) > sequence[mode_number].options[0]){
        NextMode(S);
        *swing_360_c_enabled = 0;
    } else {
        *motor_torque = 10.0 * beta_cont;
        *swing_360_c_enabled = 1;
    }
}

// ..............................................................................
// | Shake controllers |
// ..............................................................................
else if (sequence[mode_number].name == "Shake fast"){
    if ((t - t_start) > sequence[mode_number].options[0]){
        NextMode(S);
        *shake_fast_enabled = 0;
    } else {
        *motor_torque = 10.0 * beta_cont;
        *shake_fast_enabled = 1;
    }
}

else if (sequence[mode_number].name == "Shake slow"){
    if ((t - t_start) > sequence[mode_number].options[0]){
        NextMode(S);
        *shake_slow_enabled = 0;
    } else {
        *motor_torque = 10.0 * beta_cont;
        *shake_slow_enabled = 1;
    }
}

// ..............................................................................
// | No controller |
// ..............................................................................
else if (sequence[mode_number].name == "No controller"){
    if ((t - t_start) > sequence[mode_number].options[0]){
        NextMode(S);
    } else {
        *motor_torque = 0;
    }
}
APPENDIX H. SWITCHING MECHANISM CODE

else {
    motor_torque = 0;
}

static void NextMode(SimStruct *S)
{
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S, 0);
    real_T *beta_start = ssGetOutputPortRealSignal(S, 12);
    real_T *t = ssGetIPtr(S);
    real_T alpha = **uPtrs++;
    real_T alpha_dot = **uPtrs++;
    real_T beta = **uPtrs++;
    extern int mode_number;
    extern real_T t_start;
    mode_number += 1;
    t_start = *t;
    if (fabs(beta - M_PI * ceil(beta / M_PI)) < fabs(beta - M_PI * floor(beta / M_PI))) {
        *beta_start = M_PI * ceil(beta / M_PI);
    } else if (fabs(beta - M_PI * ceil(beta / M_PI)) > fabs(beta - M_PI * floor(beta / M_PI))) {
        *beta_start = M_PI * floor(beta / M_PI);
    }
}

static void mdlTerminate(SimStruct *S)
{
}

#endif

#define MATLAB_MEX_FILE
/* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#include "cg_sfun.h" /* Code generation registration function */
#endif
H.2 Sequence.h

struct node {
    char *name;
    real_T options[2];
} sequence[] = {
    "Swing up by 10 c-wise and balance with linearized controller", { 5 , 0 },
    "Free fall and balance down with linearized controller", { 2 , 0 },
    "Swing up by energy and balance with linearized controller", { 10 , 0.1 },
    "Swing down by 10 cc-wise and balance with linearized controller", { 2 , 0 },
    "Swing up by 10 cc-wise and balance with linearized controller", { 2 , 0 },
    "Swing up by energy and balance with linearized controller", { 2 , 1.4 },
    "Swing down by 10 c-wise and balance with linearized controller", { 2 , 0 },
    "Swing up by 10 cc-wise and balance with linearized controller", { 2 , 0 },
    "Swing down by 10 c-wise and balance with linearized controller", { 2 , 0 },
    "Swing up by 10 c-wise and balance with linearized controller", { 2 , 0 },
    "Swing down by 10 cc-wise and balance with linearized controller", { 2 , 0 },
    "Swing up by 10 cc-wise and balance with linearized controller", { 2 , 0 },
    "Swing 360 degrees by 10 c-wise", { 2 , 0 },
    "Swing down by 10 cc-wise and balance with linearized controller", { 5 , 0 },

    "Swing up by energy", { 2 , 1 },
    "Swing up by energy and balance with linearized controller", { 2 , 1 },
    "Free fall and balance down with linearized controller", { 2 , 0 },
    "Swing up by 10 cc-wise", { 2.5 , 0 },
    "Swing up by 10 c-wise", { 2.5 , 0 },
    "Swing up by 10 cc-wise and balance with linearized controller", { 2 , 0 },
    "Swing up by 10 c-wise and balance with linearized controller", { 2 , 0 },
    "Swing down by 10 cc-wise", { 2.5 , 0 },
    "Swing down by 10 c-wise", { 2.5 , 0 },
    "Swing down by 10 cc-wise and balance with linearized controller", { 2 , 0 },
    "Swing down by 10 c-wise and balance with linearized controller", { 2 , 0 },
    "Swing 360 degrees by 10 cc-wise", { 2 , 0 },
    "Swing 360 degrees by 10 c-wise", { 2 , 0 },
    "Shake fast", { 5 , 0 },
    "Shake slow", { 6 , 0 },
    "No controller", { -- , 0 },
};
Appendix I

Accompanying CD-ROM

With this report a CD-ROM is added which contains all files programmed and generated during this internship and are stated in the following lists.

Files for Simulations and Experiments
These files are located in the subdirectory \Furuta_Pendulum_Demo:

Furuta_control_exp.mat
Furuta_control_exp.mdl
Furuta_control_sim.mdl
Furuta_pendulum_anim.m
Furuta_pendulum_init.m
Furuta_pendulum_model.m
IO_cont_alpha.c
IO_cont_alpha.dll
IO_cont_beta.c
IO_cont_beta.dll
r3gbak.mat
rnf_part.mat
Ref3.c
sequence.h
Shake_ramp_fast.mat
Shake_ramp_fast.r3g
Shake_ramp_slow.mat
Shake_ramp_slow.r3g
Swing_360.mat
Swing_360.r3g
Swing_Down.mat
Swing_Down.r3g
Swing_Up.mat
Swing_Up.r3g
Switching_algorithm.c
Switching_algorithm.dll
Template_exp.mdl
Template_sim.mdl
APPENDIX I. ACCOMPANYING CD-ROM

Documentation files

These files are located in the subdirectory \Documentation:

Control of a rotary inverted pendulum.pdf
Db214_e.pdf
Global stabilization of a cart inverted pendulum.pdf
Inverted pendulum systems.pdf
Lyapunov stability control of pendulums.pdf
Swinging up a pendulum by energy control.pdf

Report files

These files are located in the subdirectory \Report:

app_CDROM.tex
app_datasheets.tex
app_linearization.tex
app_linearizing.tex
app_modelling.tex
app_simulink.tex
app_sim_exp_results.tex
app_smc.tex
app_source_code.tex
chapter_conclusions.tex
chapter_control.tex
chapter_identification.tex
chapter_introduction.tex
chapter_manual.tex
chapter_model.tex
chapter_setup.tex
chapter_switching.tex
Report_main.tex
Report.pdf

This directory also contains the subdirectories \Figures and \Bibliography that contain all the figures used in this report and the bibliography files respectively.

Media files

These files are located in the subdirectory \Media:

Furuta_pendulum 001.jpg
Furuta_pendulum 002.jpg
Furuta_pendulum 003.jpg
Furuta_pendulum 004.jpg
Furuta_pendulum 005.jpg
Furuta_pendulum 006.jpg
Furuta_pendulum 007.jpg
Furuta_pendulum 008.jpg
Furuta_pendulum 009.jpg
Furuta_pendulum 010.jpg
Furuta_pendulum 011.jpg
Furuta_pendulum 012.jpg
Furuta_pendulum 013.jpg
Furuta_pendulum 014.jpg
APPENDIX I. ACCOMPANYING CD-ROM

Furuta_pendulum_015.jpg
Furuta_pendulum_016.jpg
Furuta_pendulum_017.jpg
Furuta_pendulum_018.jpg
Furuta_pendulum_1.avi
Furuta_pendulum_10.avi
Furuta_pendulum_2.avi
Furuta_pendulum_3.avi
Furuta_pendulum_4.avi
Furuta_pendulum_5.avi
Furuta_pendulum_6.avi
Furuta_pendulum_7.avi
Furuta_pendulum_8.avi
Furuta_pendulum_9.avi