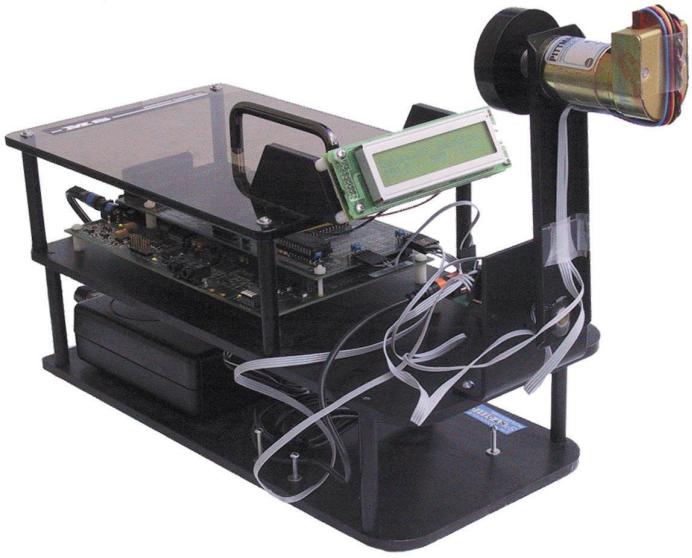
The embedded system project



Final review

2003-05-22

Abstract

As part of a project course in automatic control, we have been given the task of controlling an inverted pendulum with an inertia wheel as actuator. The task is to make the pendulum swing up from the hanging down position to balancing in the inverted position. This is a non-linear system that we linearized. We made a model and designed both LQ and PID controllers that stabilized the pendulum, but eventually settled for the LQ version. We also made a controller that can swing the pendulum up, using energy control.

Table of contents

1 Introduction

The project is part of the automatic control project course 2E1242 at KTH, and we are assigned the task of developing a control system for an inverted pendulum. This is a non-linear problem that we chose to linearize. The control law is implemented in a stand-alone digital signal processor, so that the system can function autonomously without being connected to a computer. Our task is to design a controller that can swing the pendulum up from the hanging down position to the inverted position, and then balance it there. The actuator is a motor, mounted at the end of the pendulum, that drives an inertia wheel. This report is a documentation of the whole project.

We decided to design three controllers; one for balancing the pendulum in the inverted position, one for swinging it up and one for stopping it in case of unsafe operating conditions or if the user wishes to end the demonstration. All these controllers had to be integrated to C-code that could be compiled and downloaded to the DSP.

2 The project

2.1 Resources

The project group consists of four people:

- Mikael Ek, project leader and responsible for documentation and web pages
- Michael Redmond, responsible for implementation in the DSP
- Magnus Lindhé, responsible for modelling
- Niklas Mattsson, responsible for automatic control design

The project also has an external resource, Ph.D. student Alberto Speranzon, a consultant available one hour per week in five weeks.

2.2 Time plan

The project started on the 17th of March 2003 and the final report had to be handed in before the 23rd of May 2003. We planned the work for seven weeks plus a three week break for Easter, when we wanted some time off and also needed to do some exams. The initial plan was to work sequentially, i.e. to first design a model, then a balance controller and finally an integrated version that also has the capability to swing the pendulum. Each design would then be handed to the implementation group, who implemented it. We decided on the following plan:

Milestone	Date
System modelled	2003-03-21
Balancing controller designed	2003-03-28
Swing controller designed	2003-04-02
Balancing controller in DSP	2003-04-08
Controller design ready	2003-04-08
Swing controller in DSP	2003-04-11
Safety features designed	2003-05-06
Swing and balance integrated in DSP	2003-05-06
Final controller design ready	2003-05-14
Everything implemented	2003-05-16

Table 2.1. Milestones for the project.

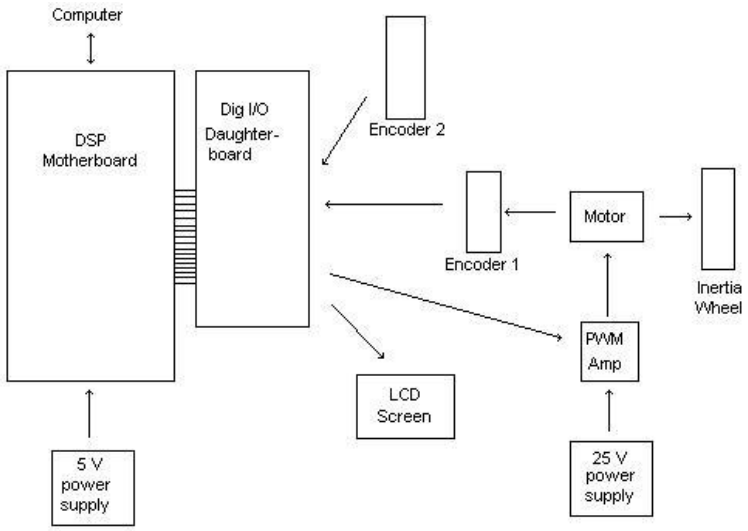


Figure 3.1. System description.

This plan was based on the assumption that it would be difficult to get to know the DSP and to implement the controller in it. Since this was easier than we thought, we got far ahead of the plan during the first two weeks. So we ended up working more in parallel, testing things right away on the real system instead of doing lengthy simulations.

3 Hardware

3.1 Description of the hardware

The embedded system we use is a control kit from Mechatronic Systems Inc. It is based on a DSP development system from Texas Instruments and, as described in figure 3.1, consists of the following parts:

- TMS320C6711 DSK board (a DSP board with parallel interface port)
- PWM/Optical Encoder data Acquisition board (a typical digital I/O board)
- PWM amplifier board
- 5V and 25V power supplies.
- 25V DC motor with 1000 counts/rev optical encoder
- Pendulum arm attached to a 1000 counts/rev optical encoder
- Inertia wheel
- A Matrix Digital LK202-25 LCD screen

There are two ways of controlling the system and those are either programming the DSP board flash memory or using the parallel port and a computer IDE, in our case Code Composer Studio. The controlling program in the DSP then uses predefined functions for the digital I/O daughter board to send a control signal to the PWM and receive data from the two optical encoders. Both the input and output are handled and translated on the daughter board. The PWM control signal goes from the daughter board to the PWM amplifier board and it is a translator between the 25V power supply and the 24V motor. The output voltage to the motor is based on the control signal from the daughter board.

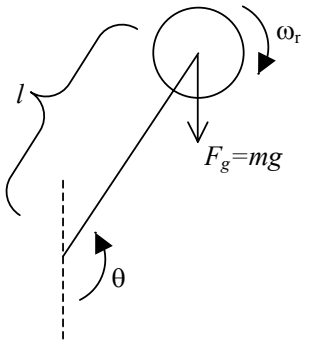


Figure 4.1.a. The pendulum notation.

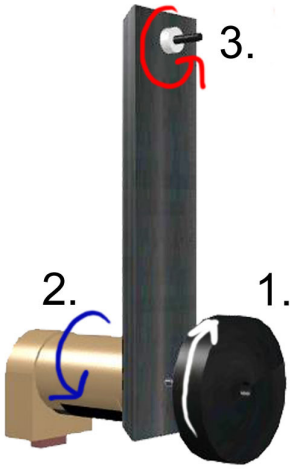


Figure 4.1.b. The actuator principle.

3.2 Getting to know the system

Our first goal was to be able to get data out of the system to create and verify our models of the system. However we had neither of the recommended software for this venture and had to come up with our own method. We decided to use the serial port of the daughter board, used by the LCD screen, and connect it directly to the computer's serial port. For this we created our own serial cable and conveniently used the daughter board's predefined function for writing to the LCD screen. The data was received and saved by a program named CodeVisionAVR, by HP InfoTech. The data was then imported to Matlab. The second step was to collect the required data for model verification and we started to examine the daughter board's encoder and PWM functions closer. We tested the inertia torque reaction when starting the motor when standing still, and the step response for the angular velocity of the motor.

4 The model

The notation for the mechanics of the pendulum is described in figure 4.1.a. We denoted the length by l, the angle of the whole pendulum by θ and the rotational velocity of the inertia wheel by ω_r. The mass m in figure 4.1.a. denotes the equivalent mass of the whole pendulum, as if it were placed at the end of the pendulum arm.

4.1 Physical effects

There are a number of physical effects that affect the movement of the pendulum. We have tried to identify the most important of these and quantify them.

4.1.1 Gravity

The mass of the pendulum arm, the motor and the inertia wheel combine to one single centre of mass, situated somewhere on the pendulum arm. It gives a gravitational torque (defined as positive in the positive θ-direction as seen in figure 4.1.a.):

$$M_g = -mgl\sin\theta,$$

where m, g and l are defined in figure 4.1.a.

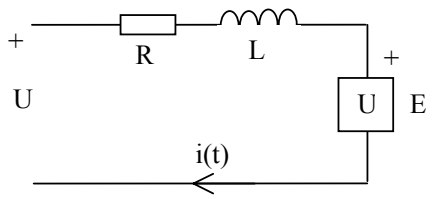


Figure 4.2. Equivalent circuit for the motor.

4.1.2 Inertia wheel

The principle of how the inertia wheel works is shown in figure 4.1.b. If the inertia wheel is to be accelerated in the direction of arrow 1, this requires a torque. An equal but opposite torque then affects the motor, as shown by arrow 2. The same torque affects the pendulum arm at the pivot, arrow 3. [1] The wheel has a mass moment of inertia, J_r:

$$M_{pend} = M_{motor} = -M_r = -J_r\dot{\omega}_r$$

4.1.3 Mass moment of inertia

When torques are applied to the pendulum or inertia wheel, they change their angle and rotational velocity according to the torque equation.

$$M = J\ddot{\theta} = J\dot{\omega}$$

This general equation holds for both J_r and J_p, J_p being the mass moment of inertia of the pendulum arm.

4.1.4 Electric characteristics of the motor

The motor can be modelled as the equivalent circuit in figure 4.2. The voltage E is the electromotorical force (EMF) that is proportional to the angular velocity of the motor. L is the inductance in the motor windings and R is a resistor that models the active power converted into heat and mechanical power. According to the data sheet of the motor, the inductance is 6.27 mH [2]. Since we have a voltage of 24 V, it would take only 0.5 ms to change the current from 0 to the maximum rated current of 2 A. This time constant is so small in comparison to the rest of the system that we can omit the inductance. There is a controller in the amplifier that controls the voltage to make the current follow the control signal *u(t)*. The torque produced by the motor can be modelled as proportional to the motor current (the reversed sign is caused by the sign convention of the ready-made software function to control the motor).

$$M_{electric} = k_i i(t) = -k_i u(t)$$

4.1.5 Friction

The pendulum is affected by both friction in the bearings and air resistance when it swings. We made a simple experiment, just letting the pendulum swing back and forth with the motor turned off. The amplitude of the swinging decreased slowly over more than a minute, so we decided that the friction there could be omitted.

The motor shaft also has some friction that has a stronger influence on the system. We modelled this friction as a reverse torque proportional to the angular velocity of the motor and inertia wheel.

$$M_{friction} = -k_f \omega_r$$

This looks very much like the effect of the EMF, and in the modelling we decided to treat it as one single breaking torque, proportional to the angular velocity.

4.2 A non-linear state-space model

We then write the sum of all the torques acting on the inertia wheel and the pendulum as

$$M_r = J_r\dot{\omega}_r = -k_i u(t) - k_f \omega_r(t)$$
$$M = J_p\ddot{\theta} = -mgl\sin\theta - M_r$$

Using three states, $\mathbf{x} = \begin{bmatrix}\dot{\theta} & \theta & \omega_r\end{bmatrix}^T$, this can be converted to a state-space model.

$$\dot{\mathbf{x}} = f(x) + g(x)u \text{ , where } \begin{cases} f(x) = \begin{bmatrix} -a\sin\theta + b_p c\omega_r & \dot{\theta} & -b_r c\omega_r \end{bmatrix}^T \\ g(x) = \begin{bmatrix} b_p & 0 & -b_r \end{bmatrix}^T \end{cases}$$

The values for these parameters can be found in [3] and are also collected in the following table.

Parameter	Formula	Value
a	$\frac{mgl}{J_p}$	78.4 s^{-2}
b_p	$\frac{k_i}{J_p}$	1.08 $V^{-1}s^{-2}$
b_r	$\frac{k_i}{J_r}$	198 $V^{-1}s^{-2}$
c	$\frac{k_f}{k_i}$	0.012 Vs

Table 4.1. Parameter values for the model.

4.3 Linearization of the model

To be able to use linear control design methods, we linearized the model around the instable equilibrium (pendulum pointing upwards). This is done by replacing the sinus term with its linear Taylor expansion at $\theta=\pi$.

$$\sin\theta = 0 + \cos\pi(\theta - \pi) + O(\theta^2)$$

The linear model then becomes

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \text{ , where } \begin{cases} \mathbf{A} = \left.\frac{\delta \mathbf{f}}{\delta \mathbf{x}}\right|_{\mathbf{x}=[0 \quad \pi \quad 0]} \\ \mathbf{B} = \mathbf{g}([0 \quad \pi \quad 0]) \end{cases}$$

Figure 4.3. Validation of motor dynamics.

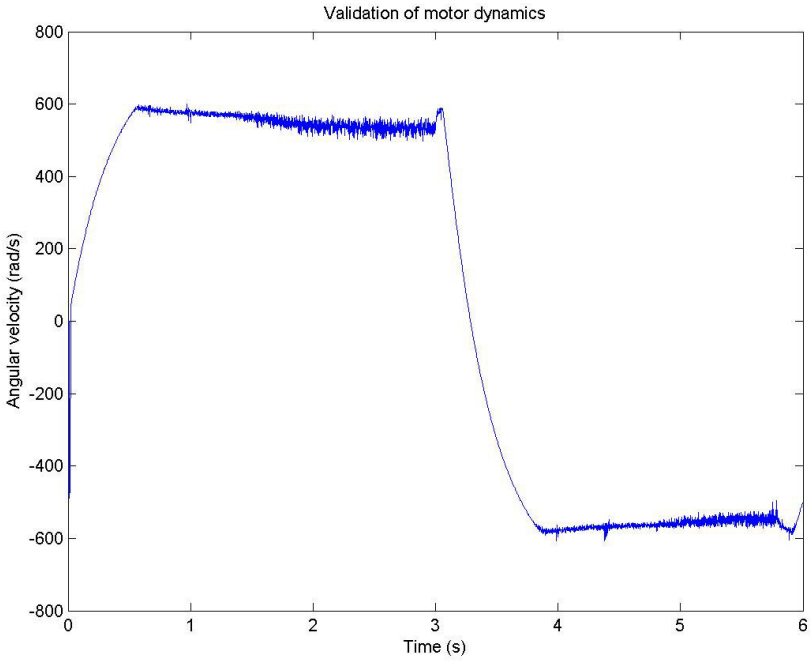


Figure 4.3. Validation of motor dynamics.

4.4 Validation of the model

As described in the section about the DSP, we managed to get data from both encoders in the Mechkit. We tried two different experiments.

4.4.1 Validation of the motor dynamics

For three seconds the motor is accelerated to full forward speed and then instantaneously switched to full reverse speed, for three more seconds. We held the motor still all the time to minimise vibrations. The result is shown in figure 4.3.

The motor dynamics agree very well with the model, except at high velocities when the velocity saturates earlier than predicted. It even decreases some, at constant current. Judging from the sound of the motor we believe that this is caused by high-frequency vibrations, possibly taking energy from the rotational movement. We have decided not to incorporate this in the model, as this would make it unnecessarily complex.

4.4.2 Validation of coupling between motor torque and pendulum swinging

This test showed the pendulum response to a step in the control signal. We started with the pendulum hanging down and the motor turned off, then at t=0.1 s we made a step to u=10. At t=5.8 s we stopped the motor again. The results are plotted in figure 4.4.

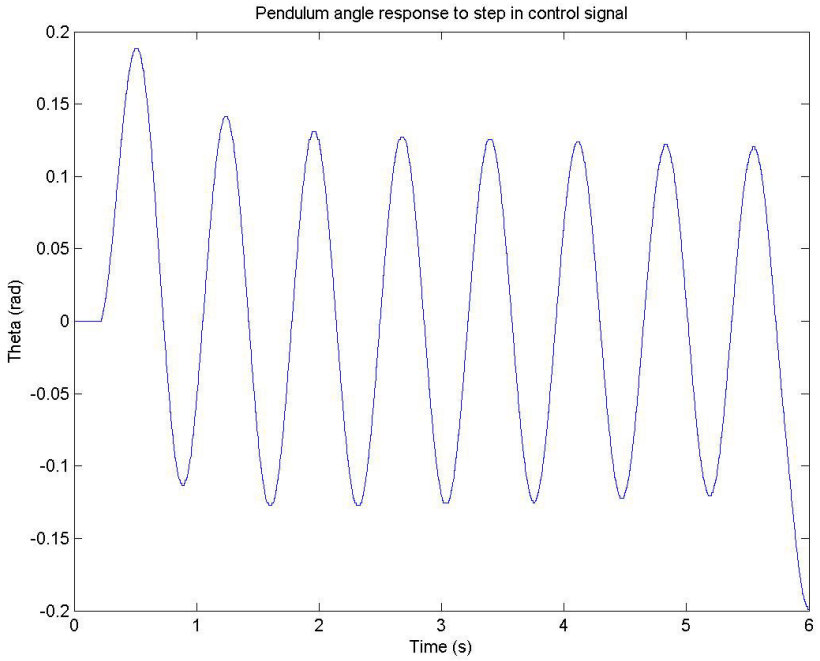


Figure 4.4. The pendulum angle response to a step in the control signal

This too agrees very well with simulations. The step response also shows that the friction in the bearings of the pendulum are negligible as the amplitude decays very slowly.

5 Controller design

To make the control design simpler we decided to divide the controller action into three different parts:

1. Balance, responsible of balancing the pendulum at the inverted equilibrium.

2. Swing, whose task is to swing the pendulum until it reaches the inverted position.

3. Stop, intended to bring the pendulum to the down position when shutting the system down. It also steps in as a safety feature if the angular velocity of the pendulum becomes too high or it has rotated more than two revolutions from the starting position. This is to avoid tearing off the cables connecting the motor and the motor encoder to the DSP.

5.1 Balance controller

We had two requirements on our controllers; we wanted them to stabilise the pendulum at the inverted equilibrium while keeping the motor velocity as low as possible. A low motor velocity gives the controller a maximal ability to react to disturbances, from any direction. If the motor has a high velocity in any direction, it will not be able to accelerate much more in that direction if needed to produce torque. When judging how good the controllers were, we tested how resistant the system was to external disturbances (poking it with a finger) and how steadily it balanced around the equilibrium when undisturbed.

5.1.1 LQ controller

Our first approach was to use an LQ controller. This approach felt appropriate to use for this kind of problem, as we wanted to minimise all the states and the control signal and make the system non-sensitive and robust. It also guarantees a phase margin of at least 60° and an infinite amplitude margin. The LQ-method minimises the integral:

$$\int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{R}u^2\right)dt, \quad \mathbf{x} = \begin{bmatrix} \dot{\theta} \\ \theta \\ \omega_r \end{bmatrix}$$

Here **Q** and **R** are constant penalty matrices. We chose **R**=100 and **Q**=**I**, meaning that all states are equally important to minimise. When simulating the system with the state feedback from the LQ it worked fine, the system stabilised at the inverted equilibrium even if the pendulum was perturbed at the start. The control law is

$$u = -\begin{bmatrix} 53.32 & 472.2 & 0.1127 \end{bmatrix}\mathbf{x}$$

When we tried to calculate the sensitivity and the complementary sensitivity, the results were strange and indicated that the controller amplified disturbancies rather than attenuated them.
Despite the bad sensitivity and the strange complementary sensitivity function we decided to make a test implementation of the controller. Just like in the simulation the implemented controller was able to stabilise the pendulum at the inverted position. The controller was even able to swing the pendulum to the inverted position and stabilise it. This was an unexpected consequence of the fact that we used a modulo operator to keep the angle $(\theta-\pi)$ in the interval $[-\pi,\pi]$. The angle thus made an abrupt change at the down position, resulting in the controller giving the pendulum impulses that made it swing. However, the controller was not able to stabilise the pendulum when the velocity was too high.

5.1.2 PID controller

After our first consultant meeting we decided to try a simpler approach than the LQG, a PID controller. We decided to use the angle θ as the input to the controller. To make sure that the motor velocity did not become too large, we also added a small proportional term, keeping it down. The first approach was to use lead-lag design directly on the instable system, but this gave strange results and a system that was theoretically stable but did not work when implemented in practice. When simulated, this controller behaved well, even when we inserted noise at the output of the process. We do not know why it did not reveal the true instability of the system.

Our second approach was to first design a PI controller, using pole placement, that theoretically stabilised the system but gave very poor performance. We then treated that closed-loop system as a new process and used the lead-lag algorithm on that. To implement this, we then integrated both the inner and outer loop into one control law. This gave reasonable results, both theoretically and practically, but still a little worse than the LQ controller.

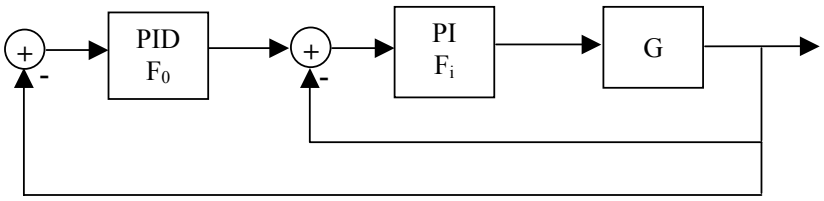


Figure 5.1. The inner and outer controllers.

The transfer functions of the inner and outer controller were

$$F_i = 100 + \frac{180}{s}, F_o = \frac{11.93\,s^3 + 280.9\,s^2 + 1891\,s + 2755}{s^3 + 74.64\,s^2 + 1386\,s}$$

When implementing the PID controller, we had some problems with integral windup. We first countered this by just resetting the integral state whenever the control signal saturated, but then we refined the approach some and just did not update the integral state when the control signal saturated. Both approaches gave good results, but the controller had some problems taking control of the system again once it had fallen out of the balancing state.

We tried using tracking, i.e. detecting when the control signal saturates, and then reducing the growth of the integral state. This requires placing the poles of both the ordinary and the saturated controller, something we never managed to to properly. So we decided that the best acheivable PID controller was with the crude form of anti-windup, and it still was not as good as the LQ.

5.2 Swing controller

There are two requirements for the swing controller: it has to swing the pendulum to the inverted position as fast as possible and it has to make sure that the pendulum enters the inverted position with sufficiently low velocity for the balance controller to be able to stabilise it. If the velocity is to low, this means that the swing controller must accelerate the pendulum and if the velocity is to high it must slow it down, e.g. if the pendulum is forcefully pushed out of the inverted equilibrium.

We first tried using a P-controller, tuned to make the downwards position unstable and thus make the pendulum swing. This gave fast swinging, but lacked control of how fast the pendulum was moving when it arrived at the inverted position.

So we instead decided on controlling the total energy of the pendulum, neglecting the energy stored in the inertia wheel. A formula for the total energy, consisting of kinetic energy, W_k, and potential energy, W_p, is given in equation 5.1.

$$W_{tot} = W_p + W_k = k_1(1 - \cos\theta) + k_2 \cdot \dot{\theta}^2 \qquad (5.1)$$

The constants k_1 and k_2 have a physical background, but the important thing for the controller is that their ratio is correct. We tuned this by setting k_1=1 and simulating the pendulum swinging with no control input. The total energy should then be constant, so we tuned k_2 to cancel out the oscillations in W_{tot}. A good value for k_2 is $6.35 \cdot 10^{-3}$.

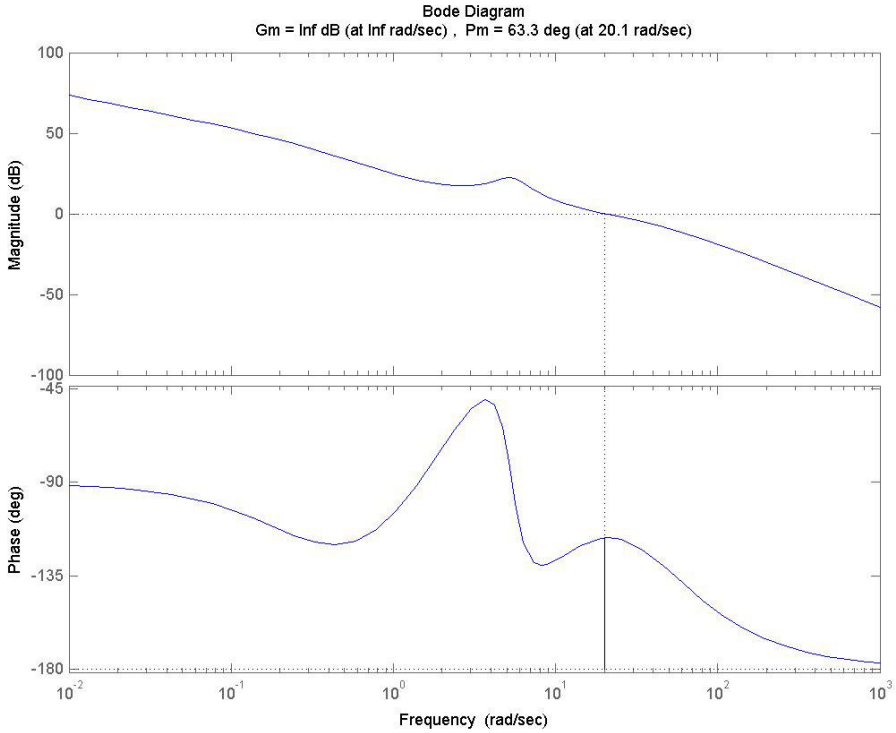


Figure 5.2. Bode diagram for the open-loop system designed with an inner PI-controller and an outer controller designed with lead-lag.

Regardless of why the pendulum enters swing mode, the goal is always to make the total energy equal to 2, i.e. the energy needed to just about reach the inverted position. The way to increase or decrease the energy of the pendulum is to accelerate the inertia wheel in the correct direction, compared to the current direction of the pendulum. This is described in table 5.1.

	Increase energy	Decrease energy
$\dot{\theta}$ *negative*	u = 10	u = -10
$\dot{\theta}$ *positive*	u = -10	u = 10

Table 5.1. Control law for the swing controller.

In simulations, the controller gave high-frequency oscillations in the control signal when the pendulum reached the inverted position and this could potentially overheat the motor. But since the balance controller has taken over by then, this is not a problem.

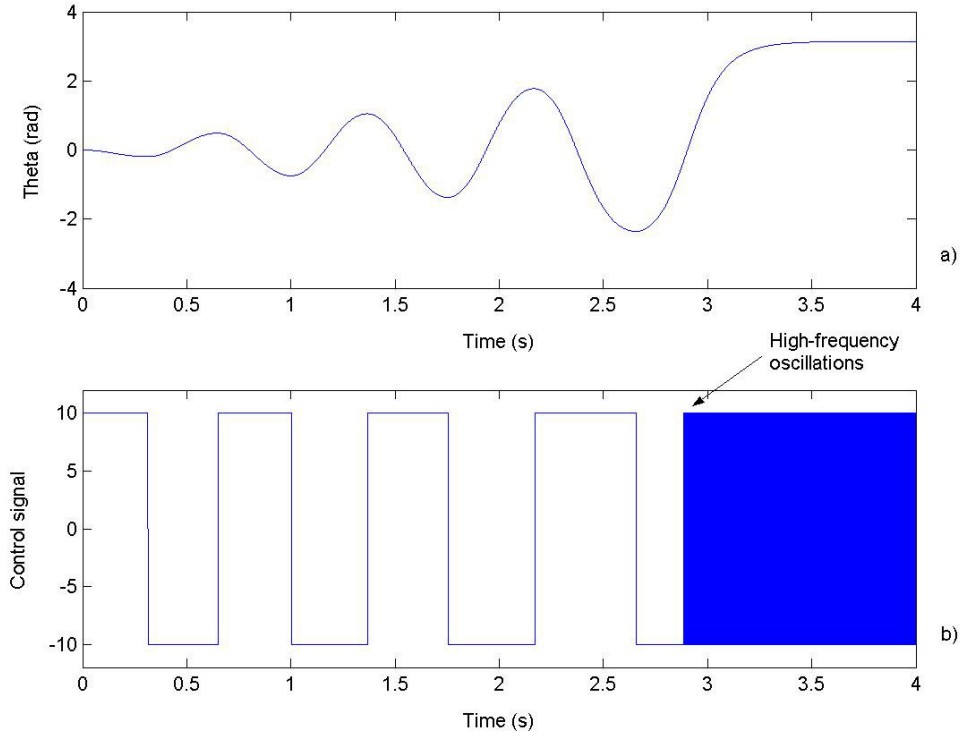


Figure 5.3. Simulation of the swing controller. Part a *shows the angle of the pendulum and part* b *shows high-frequency oscillations in the control signal as the pendulum reaches the equilibrium.*

When implemented, the swing controller made forceful swings to begin with, but the last swing was a little to weak to reach all the way. After a few weak swings, it usually reached the top. We decided that this was probably some small energy loss due to the cables, air resistance and the friction in the bearings. To compensate for this, we set the desired energy to 2.01, and the swing controller then reached the inverted position in the first trial every time! This way the pendulum always entered the inverted position with a little excess speed and, more importantly, with the inertia wheel spinning in the other direction than needed for braking the wheel. That way the controller could develop a lot of braking torque.

5.3 Stop controller

The stop controller is to be used if the velocity of the pendulum is getting to high to recover from. It will also be used when the user would like to end the balancing. We are using an LQ-controller:

$$u = -\begin{bmatrix} 277 & 25 & -0.0011 \end{bmatrix}\mathbf{x}$$

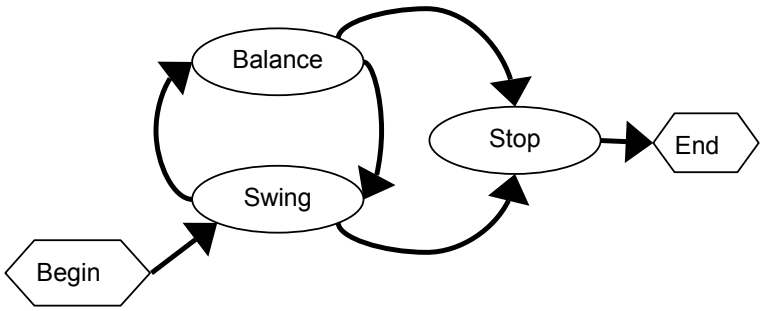


Figure 5.4. State diagram describing how the controllers are integrated.

5.4 Integrating controllers and adding safety features

When the system starts, it waits for input from the user and then starts the swing controller. The balance controller then takes over when the pendulum is 15° from the inverted position. If the pendulum falls (or is pushed) down, the swing controller takes control again. The user can stop the system by pushing a switch, making the stop controller take over. The connection between different controllers is described in figure 5.4.

The order also specified that we should add safety features that could handle situations when the speed of the pendulum becomes to high or if it has done more than two revolutions, risking to tear the cables off. If this happens, the stop controller takes over.

When we did the final testing of the system, we discovered a special situation that needed attention. If the pendulum is slowly pushed from its equilibrium, the controller tries to develop a counter-torque by increasing the speed of the inertia wheel. Eventually the speed of the wheel saturates and the pendulum falls over. The problem is then that the swing controller would need to accelerate the wheel even more in the same direction to get the required braking torque, which is impossible. We solved this by detecting the saturated condition and when it happens first set the control signal to zero for one second to slow the wheel down and then run the stop controller for another second to regain control of the pendulum and then swing it up again.

6 Implementation of controllers in the DSP

6.1 Continuous vs discrete representation

To make our calculations simpler, we decided to start by doing the controller design for continuous systems and also use a sampling frequency high enough to be able to approximate the controller with a time-continuous version. We chose a sampling time of 1 ms.

The continuous approach worked well down to a sampling frequency of 100 Hz, then the swing controller loses its ability to deliver the pendulum to the upright position with the correct energy. At about 10 Hz, the balance controller stops working as well. We therefore tried calculating a discrete version of the system and then design a discrete controller for it.

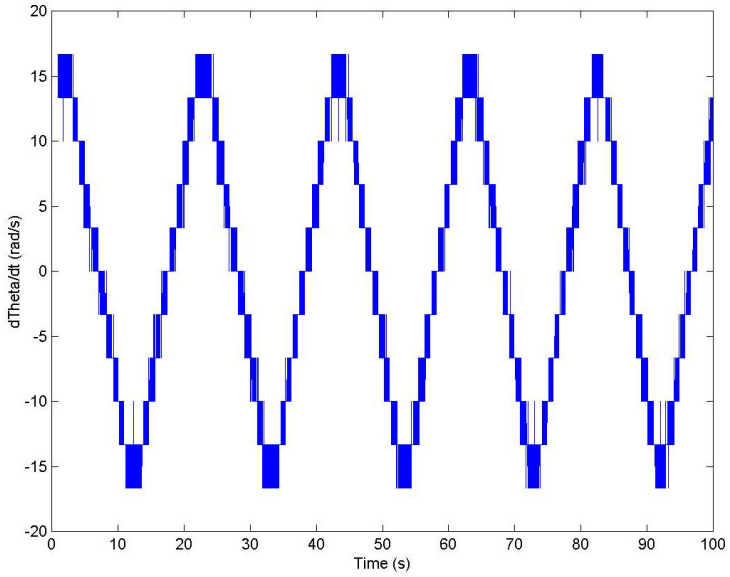


Figure 6.1. The quantisation effect when taking the difference between two consecutive angle values.

At 10 Hz the discrete model becomes (with **I** as the 3×3 identity matrix)

$$x(n+1) = \begin{bmatrix} 1.179 & 5.536 & 0.001 \\ 0.071 & 1.179 & 0 \\ 0 & 0 & 0.854 \end{bmatrix} x(n) + \begin{bmatrix} 0.071 \\ 0.002 \\ -12.208 \end{bmatrix} u(n)$$

$$y(n) = \mathbf{I}\, x(n)$$

We used LQ optimisation to find the corresponding discrete controller

$$u(n) = -\begin{bmatrix} 20.228 & 179.100 & 0.031 \end{bmatrix} x(n)$$

6.2 Differentiation

When implementing the controllers, we needed to be able to measure the derivatives of both θ and θ_r. (The encoder at the motor actually gives us values of the angle θ_r, that we then have to differentiate to get ω_r). We started by just taking the differences between two consecutive readings, but this gave a very strongly quantized derivative, due to the limited resolution of the encoders. This is shown in figure 6.1. To improve the derivative we tried a more advanced approximation [1, p. 393] that uses three old values:

$$\frac{d\theta}{dt} \approx \frac{11\theta_t - 18\theta_{t-1} + 9\theta_{t-2} - 2\theta_{t-3}}{6T}$$

This approximation has an absolute error that is proportional to T^3, which can be compared to the absolute error of the simplier one that is proportional to T. But as shown in figure 6.2 the result of this approximation is worse.

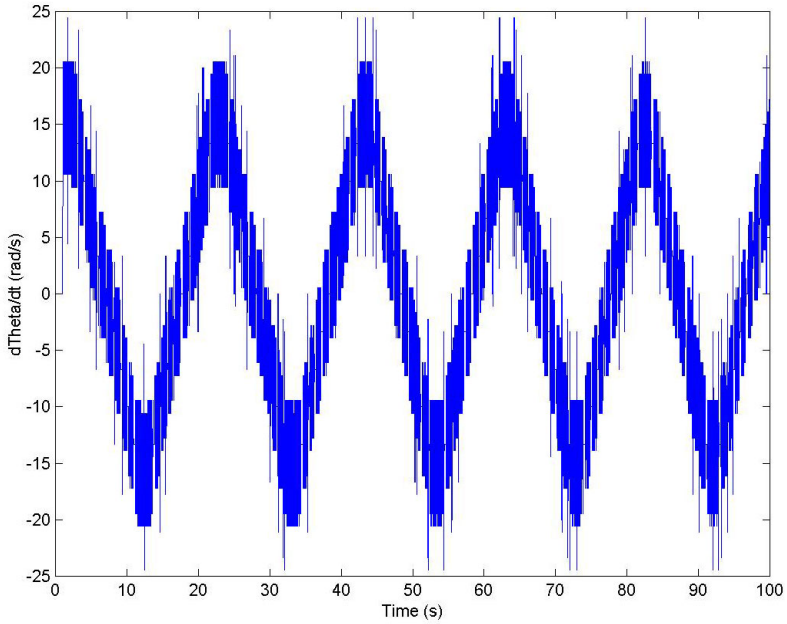


Figure 6.2. A more advanced approximation of the derivative.

The explanation to this unexpected behavior, is that the approximation is very sensitive to noise. The large coefficients in the formula amplify the already large error from the encoder and the result gets really noisy. After some simulations of the deriviative approximation in MATLAB we instead decided to use the simple approximation combined with a low pass filter, which gave a smooth derivative as in figure 6.3.

The effect of the low pass is primarily needed when the sample rate is high. When the rate is very low the filter does more harm than good because of the introduced delay. The effect of the filter is therefore decreased as the sample rate decreases and at really low rates the filter has no effect at all.

6.3 LQ controller implementation

When implementing the LQ controller we chose not to have an observer, but to differentiate the angles and treat all three states as measurable. The control law then became a simple vector scalar product.

6.4 PID controller implementation

At first we calculated the P, I and D parts respectively and then added them to get the control signal. This gave bad results, probably due to the imperfections of our derivative and integral approximations. We then converted the transfer function of the controller to state-space form, which seems more appropriate for computer implementation. This, combined with anti-windup as described in section 5.1.2, gave reasonable results.

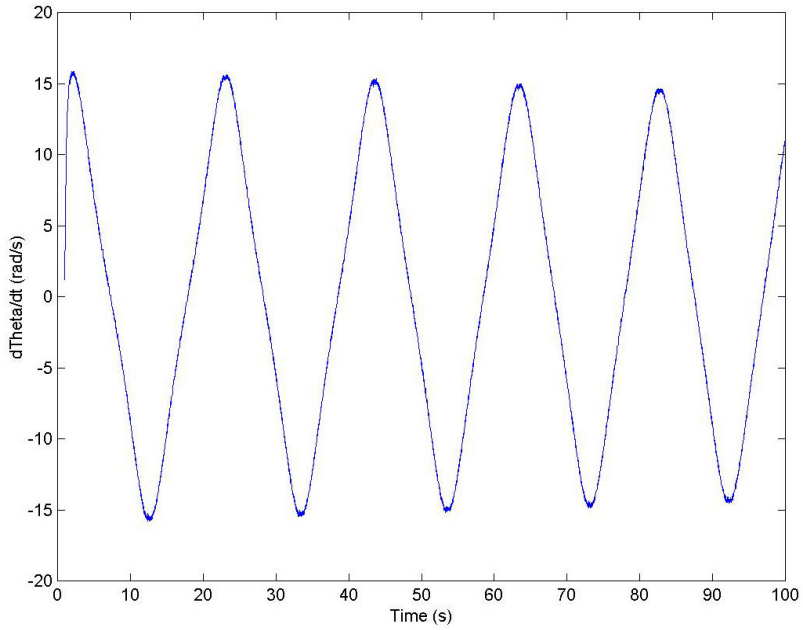


Figure 6.3. A simple difference approximation combined with a low pass filter.

7 Possible improvements

The most beneficial improvement would be a higher resolution in the encoders. This would drastically improve the differentation leading to a better prediction and determination of the concurrent speeds of both the pendulum and the inertia wheel. The more accurate measurements, i.e. inputs, to the controllers will then improve their behaviour. This will of course add to the cost of the project, so it would have to be motivated by a need of higher accuracy.

A better motor, with higher output, should also be considered as an improvement. Not so much for balancing at the inverted equillibrium but rather for the time of swinging to the upright position. A higher output from the motor gives the system more torque to utilise. A final improvement, though it might also be the hardest one to accomplish, would be to create a wireless system. The only major problem would be the motor power supply, but that could be solved by using a slip ring power connector in the axis. This would be an improvement because the wires to the encoder and motor act like an unpredictable external disturbance to the system. The rigidness of the wires has proved to be a considerable annoyance.

8 Conclusions

The work progressed faster than expected, mainly because it was easier than we thought to transfer programs into the DSP and run them. Implementing controllers has proven to be easier if they are expressed in state-space form, probably since this better matches the way a computer works. There were problems when implementing both integration (integral windup) and differentiation (quantisation noise due to limited resolution) that needed to be addressed.

LQ controllers are a user-friendly way of computing controllers for multi-variable systems, since the approach is so systematic and there are Matlab routines for most of the calculations. We tried to tune the controllers by changing the penalty matrices, but even large changes did not do much to change the performance of the system. It was also interesting to see how well a simple energy-based relay controller handled the non-linear problem of swinging the pendulum up.

9 References

1. A J Thor, A Höglund. *Partikeldynamik och statik.* Studentlitteratur, Lund, 2001.
2. Pittman motors (1999). *Datasheet: LO-COG DC servo motors*. Mechatronics Control Kit, [cd-rom], Mechatronics Systems, Inc. 2001.
3. K J Åström, D J Block, M W Spong. *The reaction wheel pendulum.* Mechatronics Control Kit, [cd-rom], Mechatronics Systems, Inc. 2001.
4. L Råde, B Westergren. *Beta Mathematics Handbook.* Studentlitteratur, Lund, 2001.