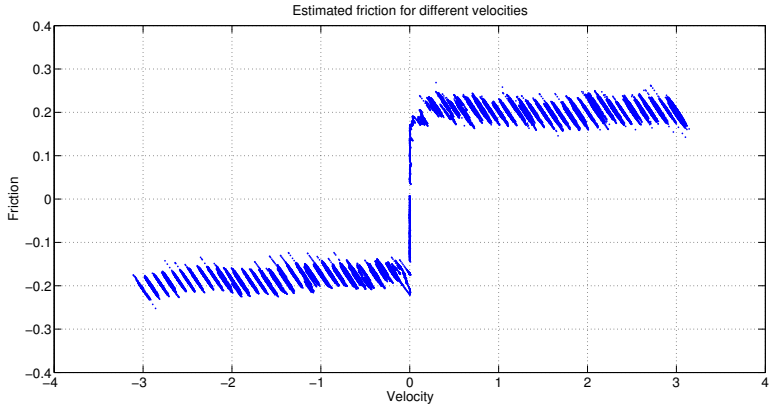
Adaptive Friction Compensation

Adam Jalkemo adam@jalkemo.se
Alexander Israelsson israelsson.alexander@gmail.com
Emil Westenius emil@westenius.se
Jonathan Andersson mat11ja1@student.lu.se

Supervisor: Gabriel Ingeson

May 24, 2016



Abstract

In this project a Furuta Pendulum has been controlled by a Java program running on a computer. The goal was to use adaptive friction compensation while balancing a pendulum in the inverted position. This was achieved by using two controllers - one Lyapunov based controller for swing up and an controller based on a Linear Quadratic Regulator, LQR, in the top position. The top controller also includes friction compensation by estimating the friction with the Recursive Least Squares, RLS, method and integral action for one state. The results show that by adding friction compensation the limit cycles in the top position is greatly reduced.

Contents

1 Introduction

In this project adaptive friction compensation of the Furuta pendulum process has been explored. The Furuta pendulum consists of a pendulum, which is free to rotate in the vertical plane, attached to the end of an horizontally rotating arm that can be controlled. The pendulum was stabilized in the inverted position using a swing up controller and a top controller. When controlling the pendulum in the top position limit cycles occurred due to friction. By compensating for the friction the pendulum was stabilized and the limit cycles reduced. The friction was estimated using an adaptive method. To do this a Java controller and interface was implemented on a linux computer.

2 Overview

This report begins with an explanation of the program structure (which is implemented in java). After that the system model used in described and the choice of controllers presented. An explanation of the resulting graphical user interface follows.

3 Program structure

To communicate with the Furuta process we use the analog box normally used during the control labs at LTH. The realtime package from Automatic Control at LTH is used for communication between the controller and the process. A general overview can be seen in fig. 1. Except for the main application thread we use one thread for the GUI and one for the controller (and three additional for the plotters, but these are not implemented by us). It was important to synchronize the controller calculations with the user inputs, we do this by using synchronized methods and blocks that java provides.

The program has three main parts: MainController, FurutaGUI and CommunicationManager. MainController has all the control logic parts, including the main control loop, object for the top controller, swingup controller and for the friction compensation. CommunicationManager is responsible for reading and passing around all the data. It reads the output from the process and handles the scaling and offset. SpecificTests is an additional class which runs tests on the process and then saves the results to files. The Discretizer class is used for the LQR parameter calculations.

The program has two main threads, the controller and the GUI. The GUI in turn has additional threads to continously update the plot windows. The controller has higher priority than the GUI threads. Since there are only two threads that access the same shared resources and only one lock active at a time there can be no deadlocks.

The objects that are shared between the controller and the GUI are the parameters and the CommunicationManager object. The GUI has its own versions

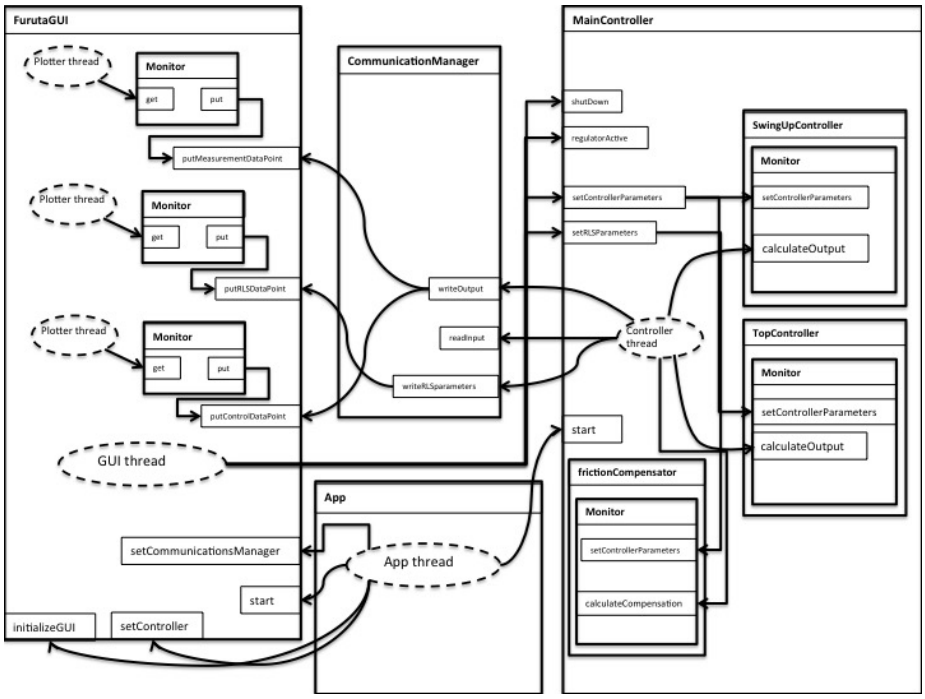


Figure 1: Overview of the program structure

of the parameter objects. The parameters are all changed by copying the parameter objects from the GUI and set in blocks that are synchronized. The CommunicationManager object is in itself thread safe and no extra precautions are needed when used in the threads.

4 System model

Due to the non-linear dynamics of the rotating arm and pendulum the modelling of the process is complicated. A model given in laboration 2 in the Non-linear control course has been used where the model has been simplified with the help of Lagrange theory. The dynamics of the model is then given by:

$$(J_p + Ml^2)(\ddot{\theta} - \dot{\varphi}^2 \sin\theta\cos\theta) + Mrl\ddot{\varphi}\cos\theta - gl(M + m/2)\sin\theta = 0$$

$$\begin{aligned} Mrl\ddot{\theta}\cos\theta - Mrl\dot{\theta}^2\sin\theta + 2(J_p + ml^2)\dot{\theta}\dot{\varphi}\sin\theta\cos\theta \\ + (J + mr^2 + Mr^2 + (J_p + ml^2)\sin^2\theta)\ddot{\varphi} = u \end{aligned} \tag{1}$$

where θ is the angle of the pendulum, φ is the angle of the arm and u is the motor torque on the arm. The states θ, $\dot{\theta}$, φ and $\dot{\varphi}$ will all be measured from the process and u is the control signal. The approximated coefficients are also taken from laboration 2 and are given by:

$$l = 0.413m \quad r = 0.235m$$

$$M = 0.01kg \quad J = 0.05kgm^2$$
$$J_p = 0.0009kgm^2 \quad m = 0.02kg$$
$$g = 9.8$$

Where l is the length of the pendulum, r is the length of the arm, M is the mass of the weight, J is moment of inerSystem.out.println(u);

4.1 Linearization

The model in eq. 1 is then linearized around

$$x = \begin{pmatrix} \theta & \dot{\theta} & \varphi & \dot{\varphi} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} \tag{2}$$

which yields

$$\dot{x} = Ax + Bu = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{bd}{ab-c^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-cd}{ab-c^2} & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{-cg}{ab-c^2} \\ 0 \\ \frac{ag}{ab-c^2} \end{pmatrix} u \tag{3}$$

where

$$a = J_p + Ml^2 \qquad b = J + Mr^2 + mr^2$$
$$c = Mrl \qquad d = lg(M + m/2)$$

This corresponds to the arm fixed at an angle corresponding to zero and the pendulum fixed in the top position. It is in this position that the friction will create oscillations and where the adaptive friction compensation will be applied.

4.2 Discretization

The linearized model should preferably be discretized using zero order hold since this is how the process is sampled. However, instead of ZOH a bilinear transformation is used in Java as an approximation of Φ since a method for calculating exponential matrices is not available in our matrix package, see equation 4. Γ is approximated using the approximation $B \cdot h$, this is not a perfect approximation but no loss in controller performance has been noted compared to using the ZOH discretized matrices. Having methods for calculating the discrete approximation of the system allows the change of sampling time online. For the most part a sample time of $h = 0.01$ have been used since this suited the system best. With $h = 0.01$ the ZOH discretization of the model yields in

$$e^{Ah} \approx (I + \frac{1}{2}Ah)(I - \frac{1}{2}Ah)^{-1} \tag{4}$$

$$x_{k+1} = A_d x_k + B_d u_k = \begin{pmatrix} 1.0016 & 0.01 & 0 & 0 \\ 0.3133 & 1.0016 & 0 & 0 \\ 0 & 0 & 1 & 0.01 \\ -0.0059 & 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} -0.0036 \\ -0.7127 \\ 0.0096 \\ 1.9125 \end{pmatrix} u_k \tag{5}$$

And with the approximation described above

$$x_{k+1} = A_d x_k + B_d u_k = \begin{pmatrix} 1.0016 & 0.01 & 0 & 0 \\ 0.3134 & 1.0016 & 0 & 0 \\ 0 & 0 & 1 & 0.01 \\ -0.0059 & 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} 0 \\ -0.7123 \\ 0 \\ 1.9125 \end{pmatrix} u_k \quad (6)$$

5 Controller

There are six measurement signal available for the controllers to use. Both the position θ and velocity $\dot{\theta}$ has two sensors each,one has higher resolution and can only be used at the top position. There is also one sensor each for the arm position φ and velocity $\dot{\varphi}$. There was no noticeable difference between the higher resolution top sensors and 360°-sensors and therefore only the two measurements from the 360°-sensors are used.

5.1 Swing up

For the swing up of the pendulum a Lyapunov based controller is used. For the Lyapunov candidate the total energy (kinetic and potential) of the pendulum is considered and given by

$$E = Mgl(cos\theta - 1) + \frac{J_p}{2}\dot{\theta}^2 \quad (7)$$

The Lyapunov candidate was chosen as $V(x, a) = E^2$. By taking the derivative of the candidate a controller $a = F(x)$ is derived which makes $\dot{V} \leq 0$ for all x. The controller will always output the maximal control signal possible in order to minimize the Lyapunov function. The resulting controller is:

$$u = sign(\cos(\theta)\dot{\theta}(cos(\theta) - 1 + \dot{\theta}^2/(2\omega^2))), \quad (8)$$

where ω is the natural frequency of the pendulum.

5.2 Top controller

In the top position LQR is used. For this the linearized model in equation (3) is used to calculate L. In order to change the cost of matrices Q and R online a way to calculate the state feedback gain vector L was implemented and given by

$$L = (R + B^T P B)^{-1}(B^T P A)$$

where P is calculated by iterating

$$P_{k-1} = A^T P_k A - (A^T P_k B)(R + B^T P_k B)^{-1}(B^T P_k A) + Q$$

backwards in time with $P_N = Q$. This solution for the Riccati problem will not work for all systems but in this case Fredrik Bagge verified that it should work.

In addition to the state feedback, feedforward is used to follow a reference value. The reference in this case is desired arm position φ. The resulting the controller has the form

$$u = -Lx + l_r r. \tag{9}$$

The factor l_r is calculated to achieve static gain 1 for reference to φ:

$$l_r = \frac{1}{C(I - \Phi + \Gamma L)^{-1}\Gamma}, \tag{10}$$

where C is chosen so that only the state φ is represented.

To remove remaining error in φ integral action is introduced. The error is integrated and multiplied by an adjustable gain and then added to the control signal. The update is as follows:

$$I_{k+1} = I_k + \frac{h}{T_i} e_k, \tag{11}$$

where T_i is a scaling factor and e_k is the error.

5.3 Controller switch

In order to switch between the two controllers advantageous conditions is necessary to enable the pendulum to be in a good position for when the top controller takes over. Since the model is linearized around 2 the pendulum needs to have both a low velocity and be close to the top for the top controller to work properly. The switching conditions is chosen as an ellipse made from $\dot{\theta}$ and θ which gives good estimations of when to switch. Since φ might be far from zero one might run into troubles if this in punished heavily. We solve this by resetting φ when the top controller takes over.

6 Friction compensation

When controlling the pendulum in the top position there are limit cycles due to the friction. To minimize these cycles RLS based friction compensation is added to the top controller. There are a lot of different ways to model friction some more complicated than others. In fig. 2 measured friction of the process can be seen for different velocities, where the measures friction is obtained by calculating the necessary control signal needed to keep the arm moving with a constant velocity. The oscillating behaviour is believed to be due to different friction in different directions. Since the average friction for the different velocities did not vary that much and the measured signals are noisy the models of the frictions were kept simple. The models used can be seen in fig. 3.

6.1 Friction estimation

To calculate the friction compensation the coefficients of a friction model is estimated using RLS. The respective models in fig. 3 will give the following

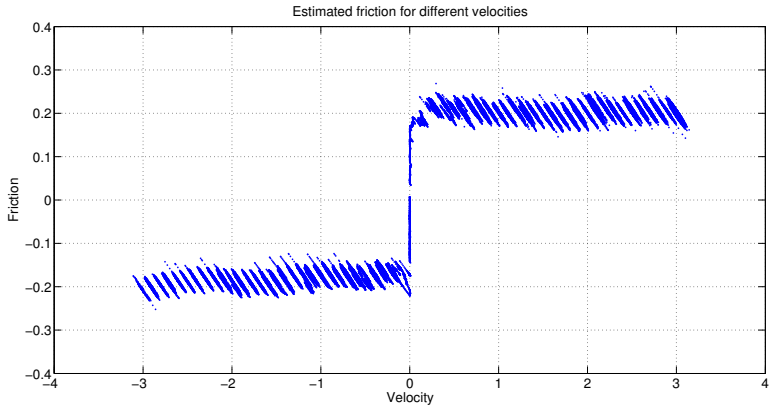


Figure 2: Measured friction of the process for different velocities. The friction is measured by using a PI-controller to keep the velocity of the arm constant with the pendulum hanging down in a fixed position. The control signal necessary for the process to keep the velocity constant can be seen as measured friction.

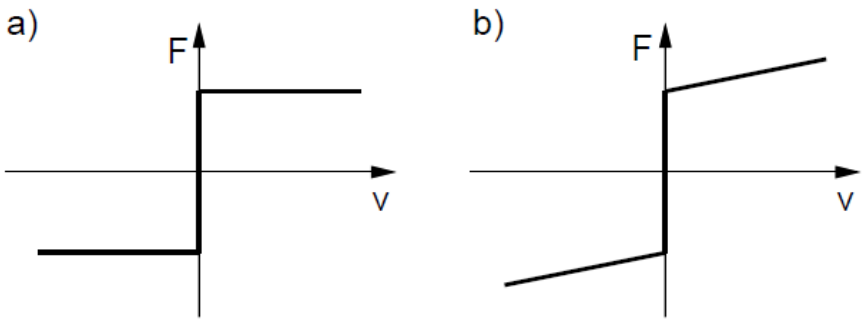


Figure 3: a) Friction model with Coulomb friction. b) Friction model with Coulomb and viscous friction. Source: `http://article.sapub.org/image/10.5923.j.mechanics.20130301.04_006.gif`

friction expressions

$$f = f_c \cdot sign(\dot{\varphi}) \tag{12}$$

$$f = f_c \cdot sign(\dot{\varphi}) + f_v \cdot \dot{\varphi} \tag{13}$$

where eq. 12 is the model in (a) and depends only on the direction of the velocity and eq. 6.1 is the model in (b) where a viscous term that also depends on the velocity is added.

The regression models for the two friction models will take the form

$$\phi = sign(\dot{\varphi}), \beta = f_c \tag{14}$$

$$\phi = \begin{bmatrix} sign(\dot{\varphi}) \\ \dot{\varphi} \end{bmatrix}, \beta = \begin{bmatrix} f_c \\ f_v \end{bmatrix} \tag{15}$$

Since the friction can be seen as a direct negative torque on the arm, it can be directly subtracted from the control signal, which using the system in eq. 6 results in

$$x_{k+1} = A_d x_k + B_d(u_k - f) \tag{16}$$

To get the RLS to work all measurements are taken one sample back in time to estimate the parameters from eq. 14 or 15. After rearranging the equation in eq. 16 we get

$$(-x_k + A_d x_{k-1} + B_d u_{k-1})/B_d = f \tag{17}$$

where f is one of the regressors in eq. 14 or 15. By choosing one of the states in x the normal RLS algorithm can be applied with some forgetting factor λ, the algorithm is given in 6.2.
Since the friction of the process have different friction for different directions of the velocity of the arm, some trials with an extra state in the regressors was also done. The extra state was added to the regressors as a constant to offset the imbalance of the friction compensation for the different directions. The regressors are then given by

$$\phi = \begin{bmatrix} sign(\dot{\varphi}) \\ 1 \end{bmatrix}, \beta = \begin{bmatrix} f_c \\ f_o \end{bmatrix} \tag{18}$$

$$\phi = \begin{bmatrix} sign(\dot{\varphi}) \\ \dot{\varphi} \\ 1 \end{bmatrix}, \beta = \begin{bmatrix} f_c \\ f_v \\ f_o \end{bmatrix} \tag{19}$$

6.2 Recursive least squares algorithm

To estimate the parameters throughout this project the standard recursive least squares algorithm with a added forgetting factor was used as stated below.

$$\hat{\beta}_k = \hat{\beta}_{k-1} + P_k \phi_k \epsilon_k \tag{20}$$

$$\epsilon_k = y_k - \phi_k^T \hat{\beta}_{k-1} \tag{21}$$

$$P_k = \frac{1}{\lambda}(P_{k-1} - \frac{P_{k-1}\phi_k \phi_k^T P_{k-1}}{\lambda + \phi_k^T P_{k-1} \phi_k}) \tag{22}$$

Above $\hat{\beta}_k$ is the estimated parameters , ϕ_k is the regressor, P_k is a matrix is the estimate of the parameter covariance and λ is the forgetting factor.

7 GUI

The operator is able to change parameters for both controllers and for the friction estimator.

For the top controller, the Q and R matrices and the integral gain can be changed. For the swing up controller, the "switching area", natural frequency,

ω, and gain can be changed. Other general parameters for the controller such as sampling time h and deadzones can also be changed.

For the friction estimation the forgetting factor λ and the initial values of θ_0 can be changed. One can also switch between the three available friction models (Coulomb friction, Coulomb with viscous friction and Coulomb with viscous friction and offset). At the top right of the GUI one can enable or disable the friction compensation.

Start and stop buttons have been added in order to record data and save to file (MATLAB .mat files are stored to the root folder of the java project). There are also tests for running step responses and for running an evaluation of the RLS convergence (without manual excitation). Both tests automatically stores the data.

The GUI will plot the outputs from the Furuta pendulum, the friction estimation and the control signal.

See fig. 4 for an image of the GUI.

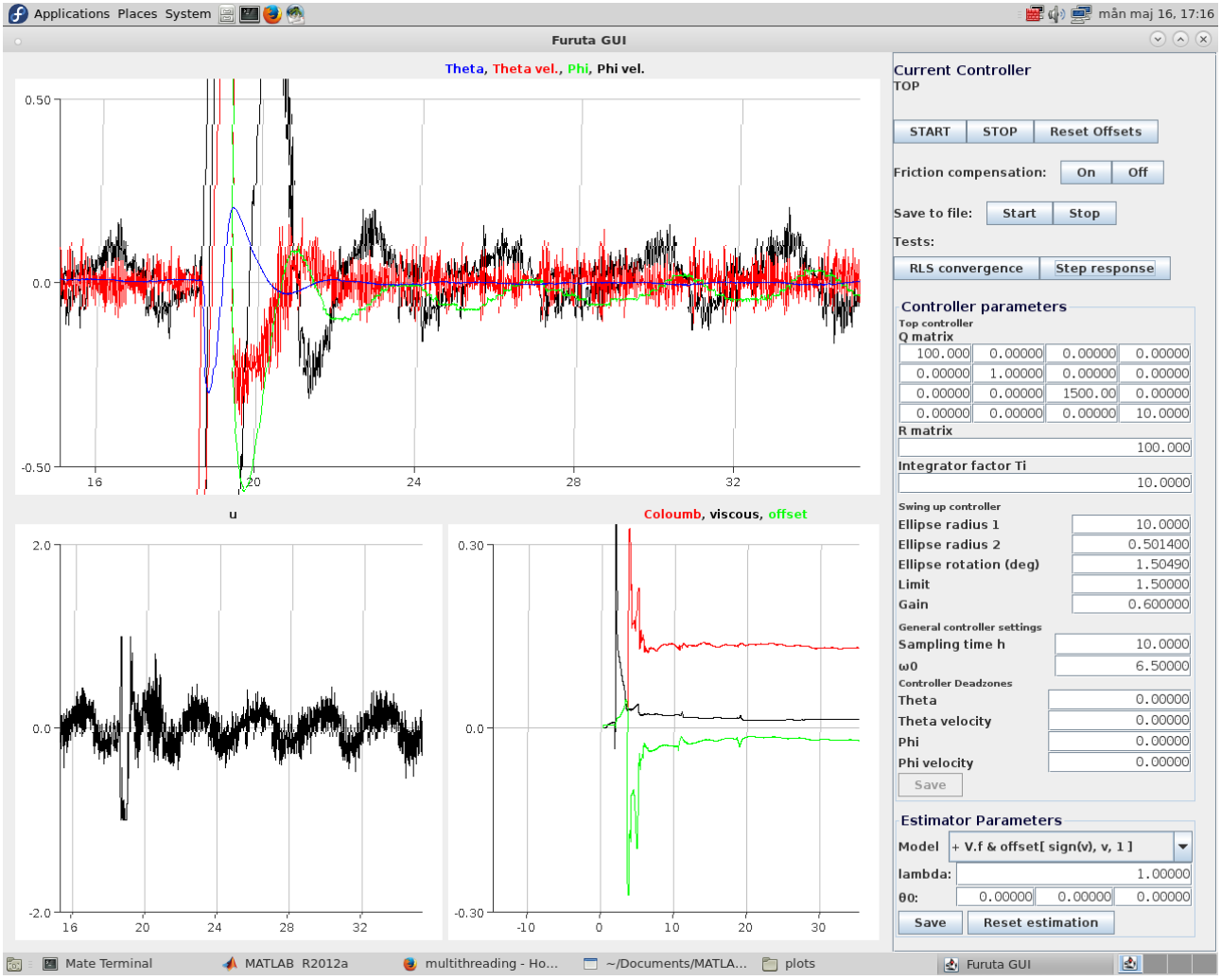
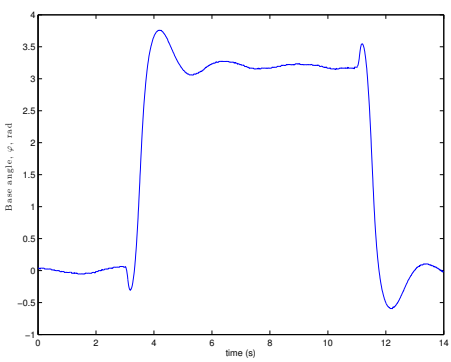


Figure 4: Overview of the program structure. There are three plotter panels in the GUI, one for the measured signals, one for the control signals and one for the RLS-parameters. There are options for changing controller parameters, friction model, RLS parameters, catching area, options to start tests and record data.

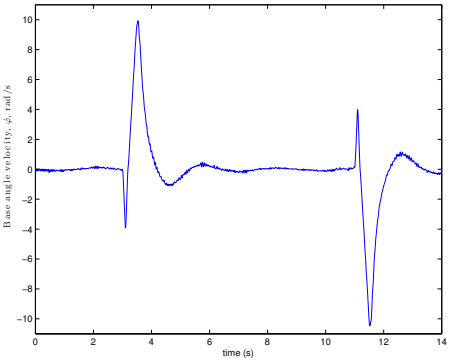
8 Results

8.1 Controller performance

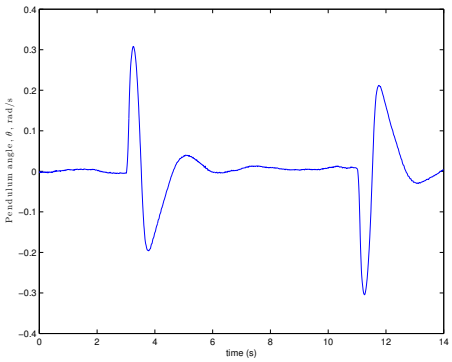
The Q and R matrices are chosen so that the error on the arm position is heavily weighted, factor 15 over both control signal and pendulum position. The velocities are relatively free in order to be able to get quick responses. This leads to a fast system with small overshoots which can be seen in fig. 5.



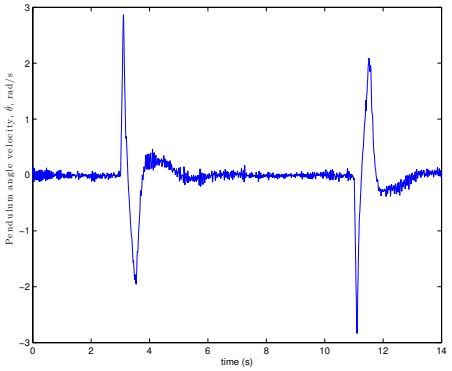
(a) The step response for the angle of the arm.



(b) The step response for the angular velocity of the arm.



(c) The step response for the angle of the arm.



(d) The step response for the angular velocity of the pendulum.

Figure 5: Step responses, π radians for the arm position φ, for the pendulum and arm positions and velocities for the using the Coulomb and viscous friction model with offset compensation and integral action.

8.2 Limit cycles

The limit cycles of the system got significantly lower when adding the friction compensation using the estimated parameters of the friction. The behaviour of the arm angle and pendulum angle can be viewed in fig. 6 where eq. 15 was used as a friction model. One can also see that the amplitude of the limit cycles are lowered to about a third of the size but that an offset in φ exists. In fig. 7 integral actions is used to counteract this offset.

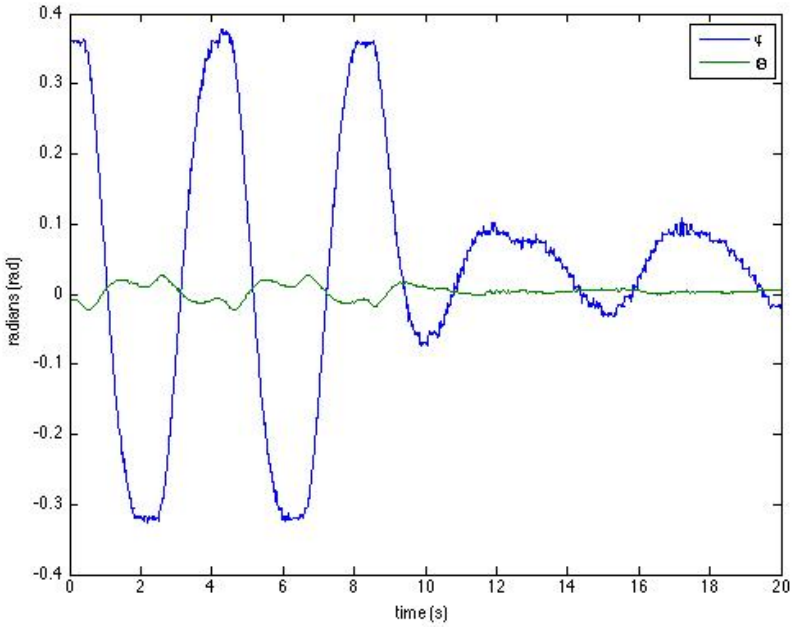


Figure 6: Arm and pendulum angle behaviour in steady state and turning on friction compensation after 10 seconds. The parameter estimation, after manual excitation (by reference changes) of the system, converged before being turned on.

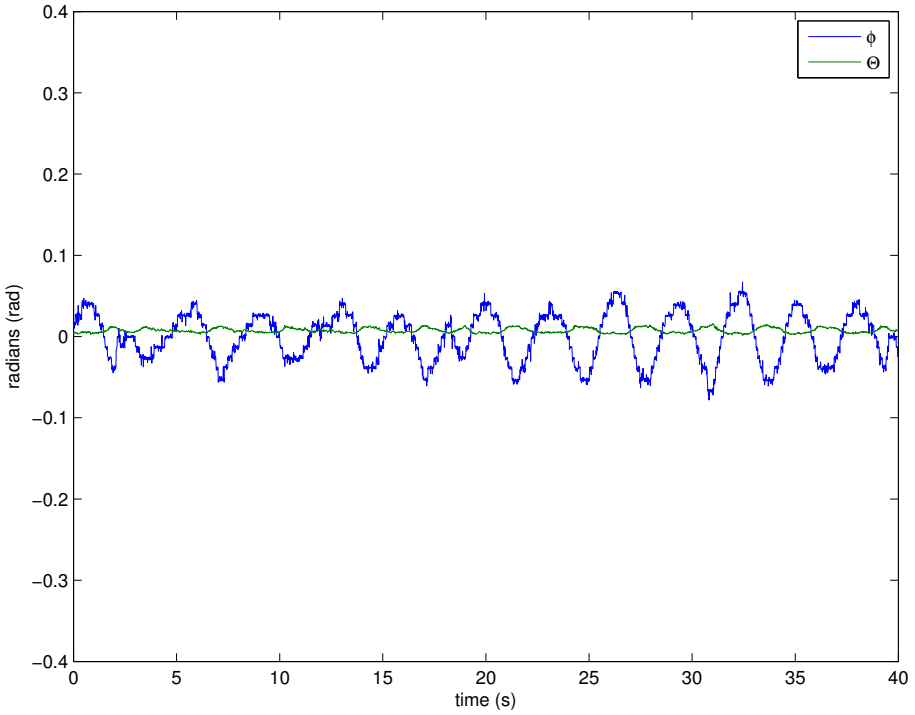
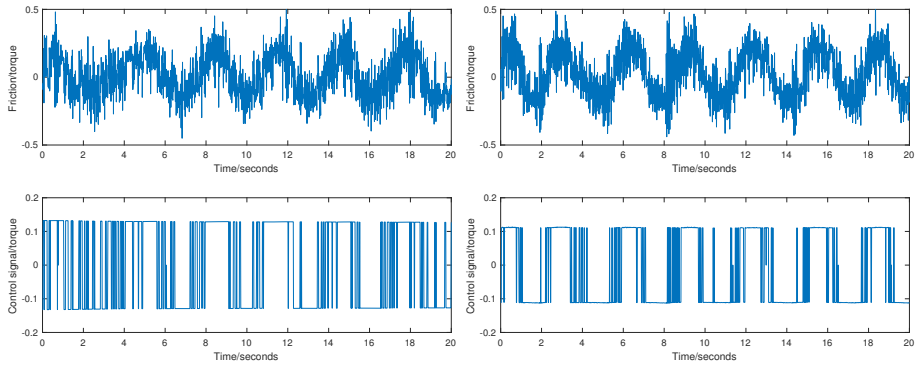


Figure 7: Arm and pendulum angle behaviour in steady state with friction offset compensation and integral action. The parameter estimation, after manual excitation (by reference changes) of the system, converged before being turned on.

8.3 Friction estimation

In fig. 8a and 8b the calculated friction and the calculated compensation signal is presented for both the Coulomb and the Coulomb and viscous friction models.



(a) The calculated friction, top, and the calculated compensation control signal, bottom, for the Coulomb friction model given in eq. 14

(b) The calculated friction, top, and the calculated compensation control signal, bottom, for the Coulomb and viscous friction model given in eq. 15. Note the similarities to 8a, i.e. the viscous part of the friction is very small.

Figure 8: Friction calculation and estimation

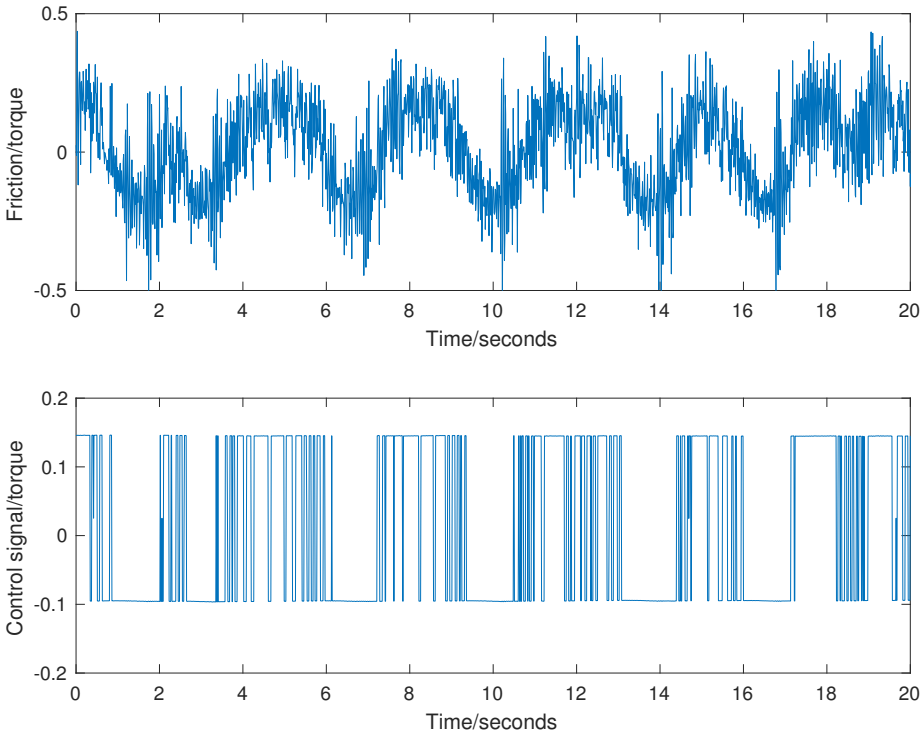


Figure 9: The calculated friction, top, and the calculated compensation control signal, bottom, for the Coulomb and viscous friction model with offset given in eq. 19. Note the offset in the control signal.

8.3.1 RLS convergence

The convergence for the parameter estimation is presented in fig. 10 both with and without external excitation through reference changes. The convergence with an added offset compensation can be seen in fig. 11

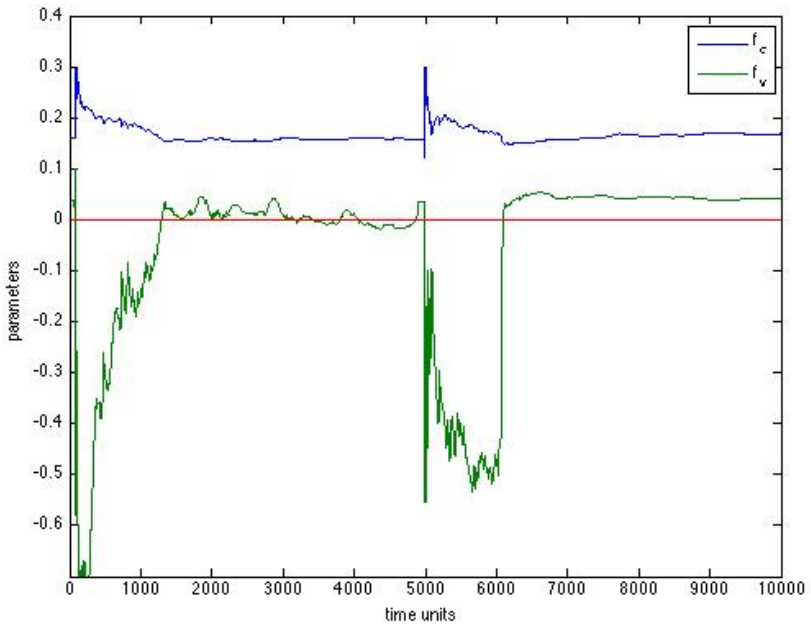


Figure 10: The convergence of the friction parameters given in eq. 15. From start up to time t=5000 the estimation was done without excitation and after t=5000 the parameters were reset and estimation was done with external excitation in the form of reference changes. A large $\lambda = 0.9999$ was used as the forgetting factor.

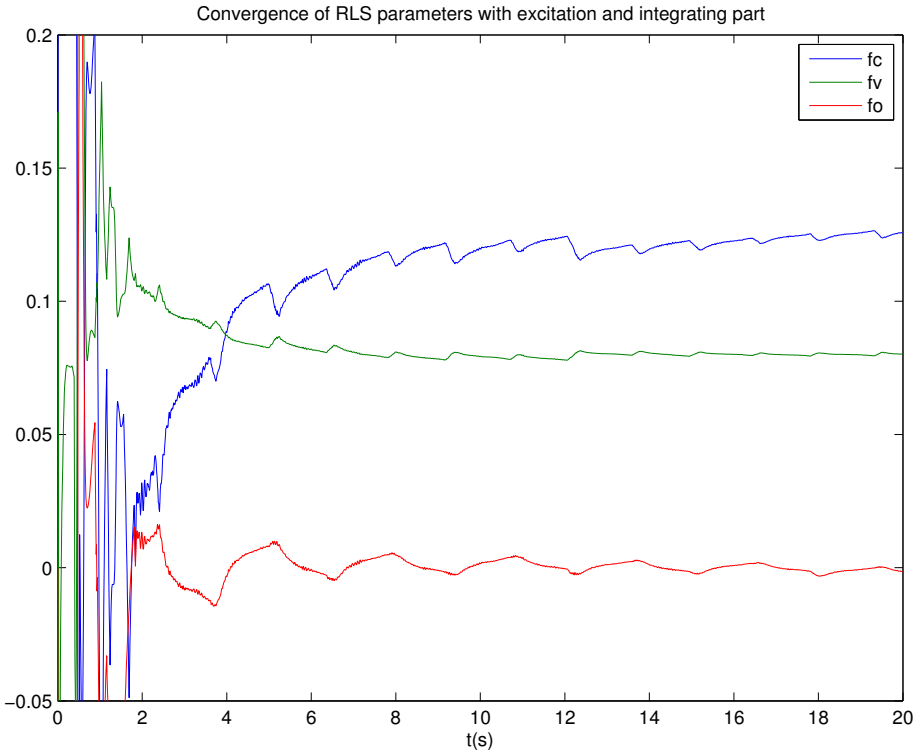


Figure 11: The convergence of the friction parameters with offset compensation given in eq. 19. The parameters converge after about 15 seconds but decent values are obtained after only about 6 seconds. The oscillating behaviour is due to multiple reference changes.

8.4 Noise

By analyzing the spectrum of the noise we could not find any disturbances affecting specific frequencies, the standard deviation σ of the signals are $\sigma_\theta = 0.209 \cdot 10^{-3}$, $\sigma_{\dot{\theta}} = 3.847 \cdot 10^{-3}$, $\sigma_\varphi = 2.282 \cdot 10^{-3}$ and $\sigma_{\dot{\varphi}} = 1.585 \cdot 10^{-3}$.

9 Discussion

9.1 Friction model

A fundamental part of this project was to select a friction model for which a good compensation could be made. As stated we chose to try a Coulomb friction model and and model with added viscous part, the difference can be seen in fig. 3. After the results in fig. 8 we decided that since they were very similar we would focus on the Coulomb with viscous friction model which had a higher potential to be close to the real friction. Since we observed that the friction is different depending on the sign of the arm velocity in fig. 2 we added a constant

offset to our regressor models, the Coulomb and viscous friction regressors with and without offset can seen in equation eq. 15 and 19 respectively. This leads to a better representation of the friction which can be seen in fig. 9. The discrepancy in the amplitude of the friction calculation is likely due to process changes, however fig. 9 still shows the better representation of the calculated friction.

9.2 Offset in φ

Our initial thought was that the different frictions depending on the sign of base angular velocity would be the reason for the offset we can see in fig. 6. However after adding the offset we could still observe the same behaviour. There was a noticeable change in the movement of the arm which can be seen in figure 9 compared to the cycles in for instance 8b. The arm stayed close to the origin for a longer period of time before it drifted away as a result of a not perfect controller (as seen by the heavy switching of the sign of the friction compensation). Since we set out to decrease the offset in φ we did our best to make sure that any offsets in the measurements are minimized and explored the option to add integral action on the state φ in order to move the arm position so it would oscillate around the true zero position instead, the result of adding this integrative part was that the offset decreased and the result can be seen in fig. 7.

9.3 RLS

As expected the convergence for the RLS friction coefficients are a lot better with the external excitation compared to when there are no external excitation at all. Without the extra excitation the coefficients would sometimes not converge at all or even converge to negative values. With the external excitation we got a pretty fast convergence to values that would be considered reasonable for the coefficients. The external excitation was simply done by not using the compensation to begin with, resulting in the limit cycles exciting the the system enough to get a good convergence of the coefficients.

As for the forgetting factor we tested some different values and found that using a high value $\lambda = 0.9999$ works the best since out signals are quite noisy and there should not be any large changes to the friction or the process.
As we can see the limit cycles does not disappear completely. This can depend on a number of different factors, one major factor is likely that the offset for the pendulum position in the top position is not perfect which would lead to over- and under correction i.e. the controller will operate on position values which are not true. Another thing is that the signals are quite noisy so it is hard for the controller to stabilize around the zero position since it keeps jumping from positive to negative due to the noise. To deal with the noisy signals trials with a implemented Kalman filter was done but the results were not very good since the true position of the arm around the zero is still not represented correctly. For when the arm gets stationary in a position there will also be stiction friction

to consider, this will increase the oscillations since it friction will be larger in this situation.

9.4 Noise

Looking at the measurement signals it is apparent that the standard deviation on especially φ affects how well both the LQR controller and the friction compensation will operate around the origin, since there is always a risk of generating a control signal in the wrong direction due to the noise. Another reason for why there might be an issue with trusting the velocities too much is that they are not directly measured by the process but instead calculated using the position. To avoid generating a control signal in the wrong direction and also avoid a noisy control signal we experimented with deadzones around the origin. This did however not increase the performance but instead increased the limit cycles. Our theory is that the somewhat noisy control signals aren't as bad as disabling the friction compensation and/or the LQR controller around a deadzone, partly due to the pendulum vibrating thus decreasing friction and our rather fast sampling time actively counteracting too much deviation.

9.5 Real-time implementation in java

Our implemented controllers work as expected in Java and threads shares resources in without blocking being a factor. The controller works for a sample time down to 4ms but since the process most likely is designed with analog filters to avoid aliasing we stuck to using a sample time of 10ms (which is used in laboratory exercises) since we didn't notice any improvements when using a lower sampling time. The GUI thread does not block the controller yet the GUI is responsive. When we run the built in tests we don't allow user input and therefore it was easiest to implement it by putting the thread to sleep (which gives somewhat of a bad user experience) until the test is finished.

10 Conclusion

The project goal was to use adaptive friction compensation while balancing the Furuta Pendulum in an inverted position. This was successfully done by using two separate controllers - one for swing up and one for the top position. The top position controller has, in addition to being an LQR, adaptive friction compensation, which greatly reduces the limit cycles, and integral action which removes the stationary error. The resulting controller stabilizes the pendulum in the top position with very small limit cycles, responds quickly to step changes and can manage reasonable disturbances.