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 16, 2016

1 Introduction

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

2 Program structure

For our suggested solution we will need one thread for the GUI and one for the controller. To communicate with the Furuta process we will use the analog box normally used during the control labs at LTH. The realtime package from Automatic Control at LTH will be used for communication in the GUI and controller implementation. It will be important to synchronize the controller calculations with the user inputs. A general overview can be seen in figure 1.

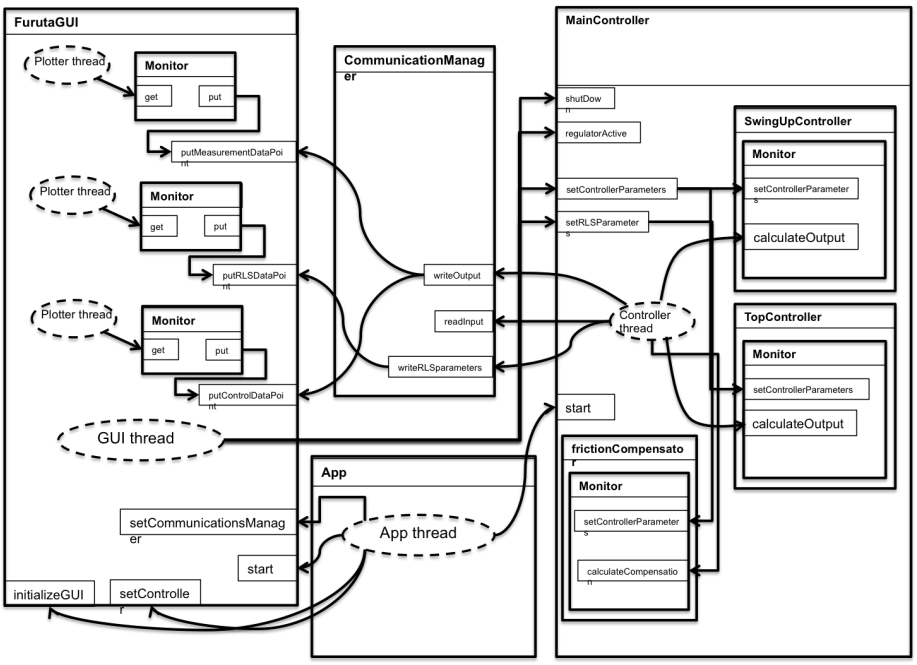


Figure 1: Overview of the program structure

The program has three main parts: MainController, FurutaGUI and CommunicationManager. The MainController has all the control logic parts. This includes the main control loop, object for top control, object for swingup con-

trol and an object for friction compensation. The CommunicationManager is responsible for reading and passing around all the data. It reads the output from the process and fixes 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 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.

3 System model

Due to the non-linear dynamics of the rotating arm and pendulum the modelling of the process becomes complicated. A model given in Lab2 in "Non-linear control" has been used where the model have been simplified with the help of Lagrange theory. The dynamics of the model is then given by:

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

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

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 will be the control signal. The approximated coefficients are also taken from Lab2 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 inertia for pendulum, J_p is the moment of inertia for the arm, m is the mass of the pendulum and g is the gravity constant.

3.1 Linearization

The model in 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.

3.2 Discretization

The linearized model should preferably be discretized using ZoH. However, instead of ZoH a bilinear transformation is used in Java as an approximation of Φ since exponential matrix was not available in our matrix package. This allows the change of sampling time online. Γ is approximated using the approximation B*h, this is not a perfect approximation but we've not notices any major losses compared to using the ZoH discretized matrices. 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

$$x_{k+1} = A_d x_k + B_d u_k = \begin{pmatrix} 1.0016 & 0.01 & 0 & 0 \\ 0.3133 & 1.0016 & 0 & 0 \\ 0 & 0 & 1 & 0.01 \\ -0.0059 & 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} -0.0036 \\ -0.7127 \\ 0.0096 \\ 1.9125 \end{pmatrix} u_k \tag{4}$$

And with the approximation described above

$$x_{k+1} = A_d x_k + B_d u_k = \begin{pmatrix} 1.0016 & 0.01 & 0 & 0 \\ 0.3134 & 1.0016 & 0 & 0 \\ 0 & 0 & 1 & 0.01 \\ -0.0059 & 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} 0 \\ -0.7123 \\ 0 \\ 1.9125 \end{pmatrix} u_k \tag{5}$$

4 Controller

There are six measurement signal available for the controllers to use. Two for the position θ and velocity $\dot{\theta}$ of the pendulum each (one has higher resolution and can only be used at the top position) and one for the arm position φ and velocity $\dot{\varphi}$. Monitoring the signals there was no noticeable difference between the higher resolution and 360°-sensor and we therefore only used the two measurements from the 360° sensor.

4.1 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 will be considered and is given by

$$E = Mgl(cos\theta - 1) + \frac{J_p}{2}\dot{\theta}^2 \tag{6}$$

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

4.2 Top controller

In the top position a LQR controller is used. For this the linearized model in eq. 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

$$L = (R + B^T PB)^{-1}(B^T PA)$$

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.

To remove remaining error in ϕ an integral action is introduced. The error is simply integrated and multiplied by an adjustable gain and then added to the control signal. Anti-Windup is done with a resetting factor.

4.3 Controller switch

In order to switch between the two controllers good 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

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

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 will give 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.

5 Friction compensation

When controlling the pendulum in the top position there will be limit cycles due to the friction. To minimize these cycles RLS based friction compensation is added to the top controller. When the pendulum is close to the top position a dead-zone is added in order to keep the noise from the measurements from making the friction compensation add control signal in the wrong direction. There are a lot of different ways to model friction. Some really complicated and some pretty simple. In fig 2 measured friction of the process can be seen for different velocities. 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.

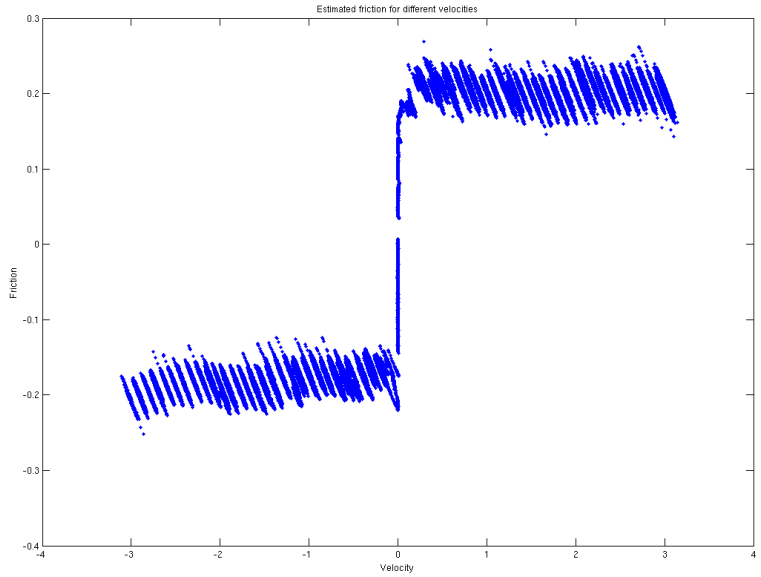


Figure 2: Measured friction of the process for different velocities

5.1 Friction estimation

To manage the friction an estimate of a simple friction model is derived using RLS. The models in fig 3 will respectively give the following friction expressions

$$f = f_c \cdot sign(\dot{\varphi}) \tag{7}$$

$$f = f_c \cdot sign(\dot{\varphi}) + f_v \cdot \dot{\varphi} \tag{8}$$

where 7 is model a and only depends on the direction of the velocities and 8 is the model in b where a viscous term that depends on the velocity is added.

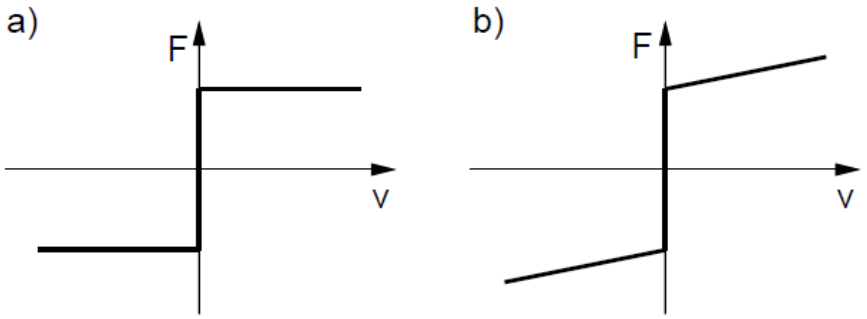


Figure 3: a) Friction model with coulomb friction. b) Friction model with coulomb and viscous friction.

The regression models for the two friction models will take the form of 9 and 10 respectively.

$$\phi = sign(\dot{\varphi}), \beta = f_c \tag{9}$$

$$\phi = \begin{bmatrix} sign(\dot{\varphi}) \\ \dot{\varphi} \end{bmatrix}, \beta = \begin{bmatrix} f_c \\ f_v \end{bmatrix} \tag{10}$$

Since the friction can be seen as a direct negative torque on the arm, it can be directly subtracted from the control signal which gives us the system in 5

$$x_{k+1} = A_d x_k + B_d(u_k - f) \tag{11}$$

To get the RLS to work all measurements are taken one step back in time to estimate the parameters from 9 or 10. After rearranging the equation in 11 we get

$$\frac{x_k - A_d x_{k-1} - B_d u_{k-1}}{-B_d} = f \tag{12}$$

where f is one of the regressors in 9 or 10. 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 5.2.
Since the friction of the process have different friction for different directions 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

$$\phi = \begin{bmatrix} sign(\dot{\varphi}) \\ 1 \end{bmatrix}, \beta = \begin{bmatrix} f_c \\ f_o \end{bmatrix} \tag{13}$$

$$\phi = \begin{bmatrix} sign(\dot{\varphi}) \\ \dot{\varphi} \\ 1 \end{bmatrix}, \beta = \begin{bmatrix} f_c \\ f_v \\ f_o \end{bmatrix} \tag{14}$$

5.2 Recursive least squares algorithm

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

$$\hat{\theta}_k = \hat{\theta}_{k-1} + P_k \phi_k \epsilon_k \tag{15}$$

$$\epsilon_k = y_k - \phi_k^T \hat{\theta}_{k-1} \tag{16}$$

$$P_k = \frac{1}{\lambda}(P_{k-1} - \frac{P_{k-1}\phi_k \phi_k^T P_{k-1}}{\lambda + \phi_k^T P_{k-1} \phi_k}) \tag{17}$$

Above $\hat{\theta}_k$ 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.

6 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 integrating factor can be changed. For the swing up controller, the "switching area", natural frequency 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 ω_0 can be changed. One can also switch between the three available friction models (Coloumb friction, viscous friction and viscous friction with offset). At the top right of the GUI one can enable or disable the friction compensation.

In order to save data to file buttons have been added to start and stop this process (MATLAB .mat files are stored to the root folder of the java project). There are also tests for running step responses and for running a test 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 figure 4 for an image of the GUI.

7 Results

7.1 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 5 where 10 was used as a friction model.

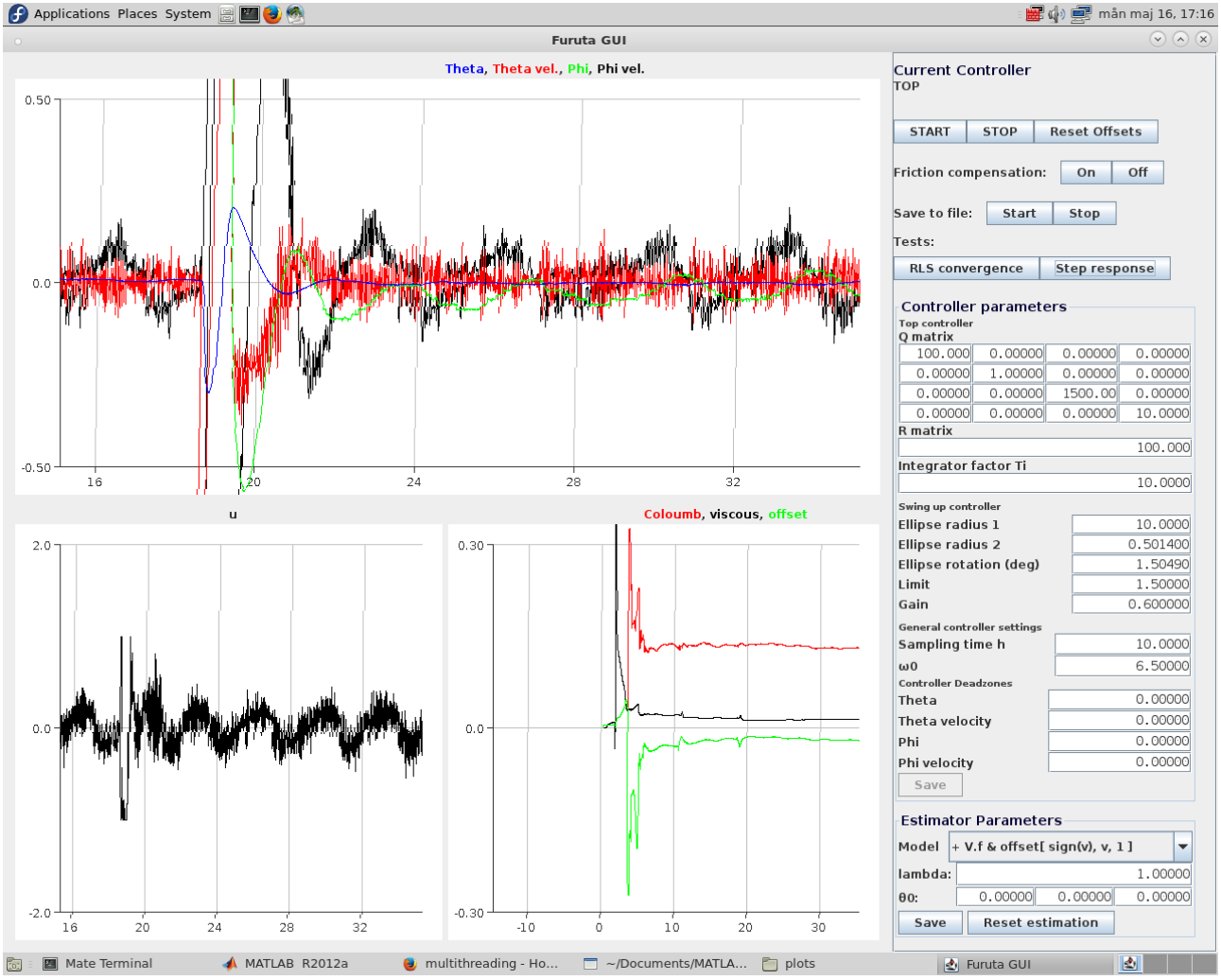


Figure 4: Overview of the program structure

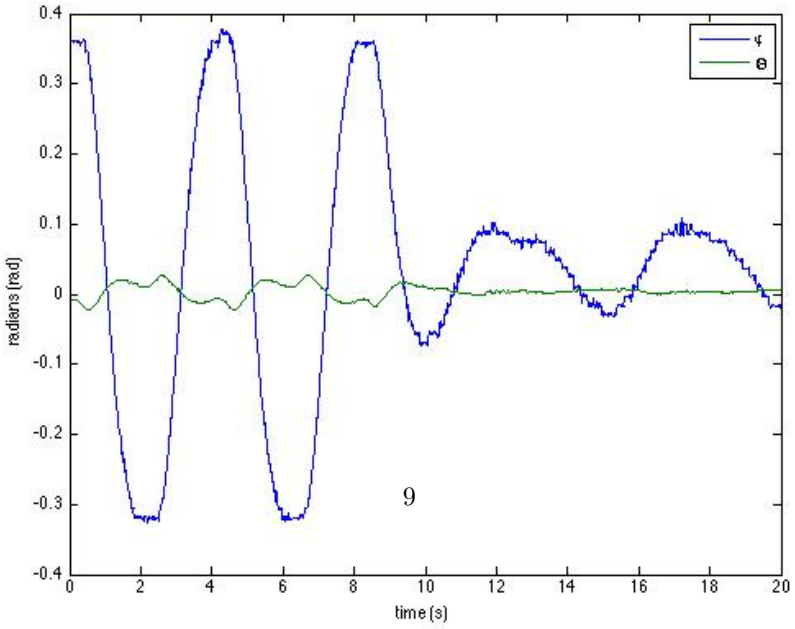
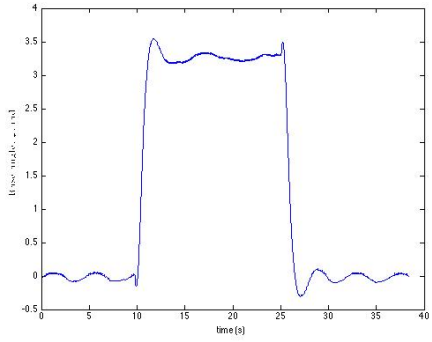


Figure 5: Arm and pendulum angle behaviour in steady state and turning on friction compensation after 10 seconds. The parameter estimation, after manual

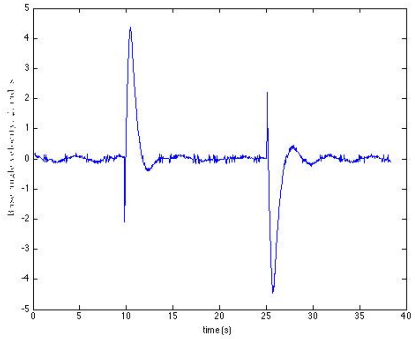
In 5 see that the amplitude of the limit cycles are lowered to about a third of the size. We can also see an offset from the zero angle.

7.2 Coloumb and viscous friction model

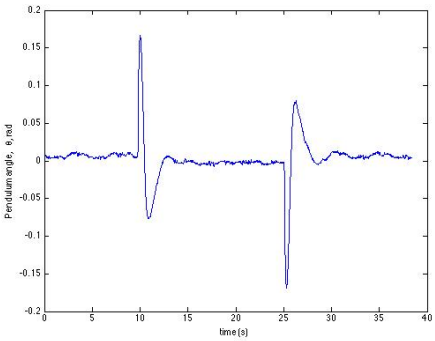
In 6 we can see the behaviour of the angle and angular velocity for both the arm and the pendulum.



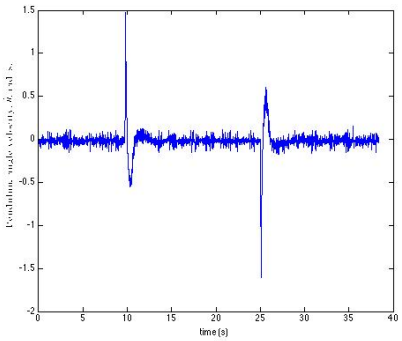
(a) The step response for the angle of the arm.



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



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



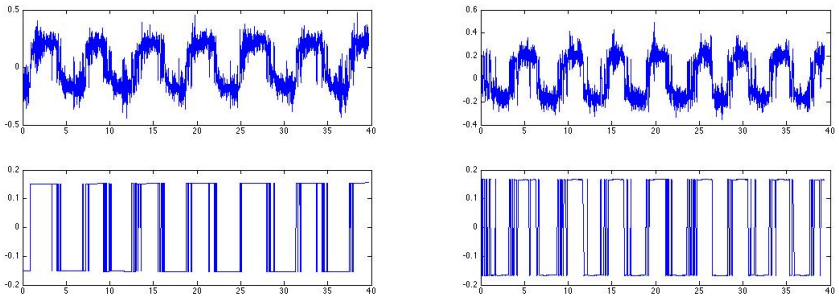
(d) The step response for the angular velocity of the pendulum.

Figure 6: Stepresonses for the pendulum and arm positions and velocities for the using the coloumb and viscouse friction model.

7.3 Friction estimation

In figure 7a and 7b the calculated fricion and the calculated compensation signal is presented for both the coloumb and the coloumb and viscouse friction models.

Figure 8: The calculated friction and the calculated compensation control signal for the coloumb and viscouse friction model with offset given in eq 14



(a) The calculated friction and the calculated compensation control signal for the coloumb friction model given in eq 9

(b) The calculated friction and the calculated compensation control signal for the coloumb and viscouse friction model given in eq 10

Figure 7: Friction calculation and estimation

7.4 RLS estimation

The convergence for the parameter estimation is presented in figure 9 both with and without external excitation.

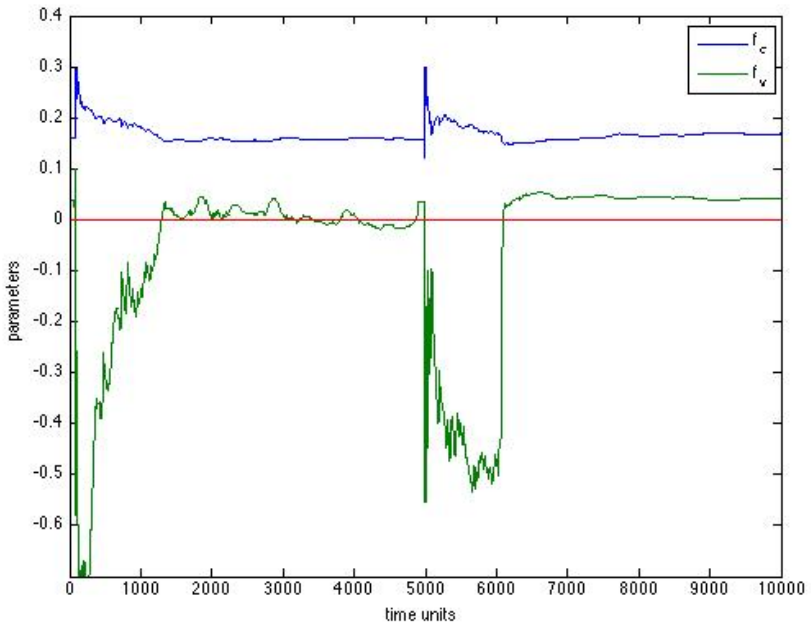


Figure 9: The convergence of the friction parameters given in eq 10. At time t=5000 time units the estimation is reset and the system is subjected to external excitation.

8 Discussion

8.1 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 figure 3. After the results in 7 we decided that since they were very similar we would focus on the coulomb and 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 2 we added a constant offset to our regressor models, the coulomb and viscous friction regressors with and without offset can seen in equation 10 and 14 respectively. This leads to a better representation of the friction which can be seen in figure 8. The discrepancy in the amplitude of the friction calculation is likely due to process changes, however 8 still shows the better representation of the calculated friction.

8.2

Our initial thought was that this would be the reason for the offset we can see in 5 however after adding the offset we could still observe the same behaviour with only a small improvement, see **??**. Instead we explored the option to add an integrative part 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 **??**.

This resulted in an improved limit cycle behaviour and marginally better step responses.

8.3 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 Conclusion