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 a Furuta pendulum process will be used. 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 we aim to stabilize the pendulum, the friction will be estimated using an adaptive method. To do this a Java controller and interface will be implemented and run 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.

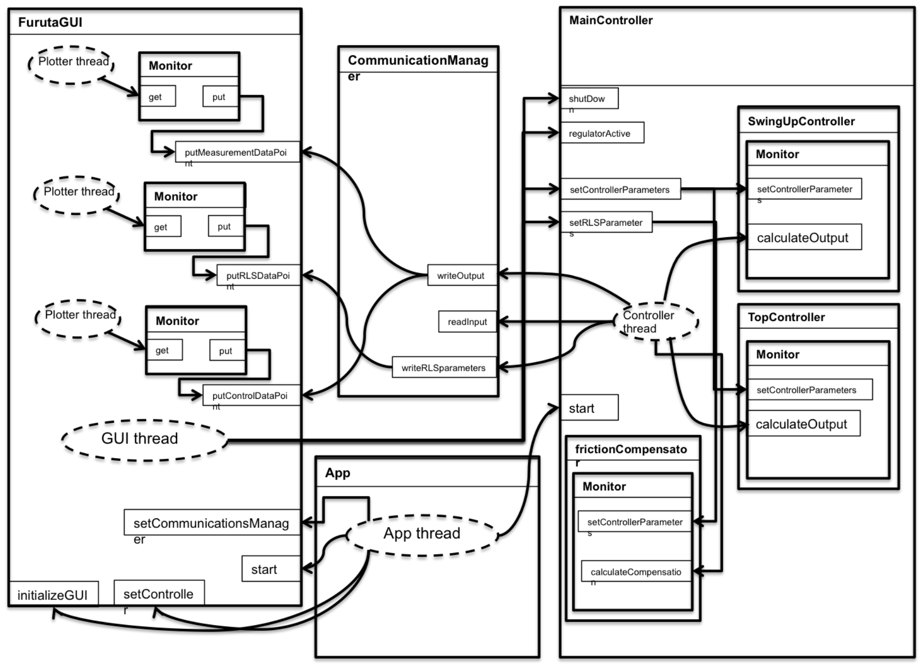


Figure 1: Overview of the program structure

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 control 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. We have used the model given in Lab2 in "Non-linear control" 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 our 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 is then discretized using ZoH. A simplified ZoH method have been implemented in java which makes it possible to change the sample 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

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

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 is used at the top position) and one for the arm position φ and velocity $\dot{\varphi}$.

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

$$E = Mgl(cos\theta - 1) + \frac{J_p}{2}\dot{\theta}^2 \tag{5}$$

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

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

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.

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.

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.

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

$$f = f_c \cdot sign(\dot{\varphi}) \tag{6}$$

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

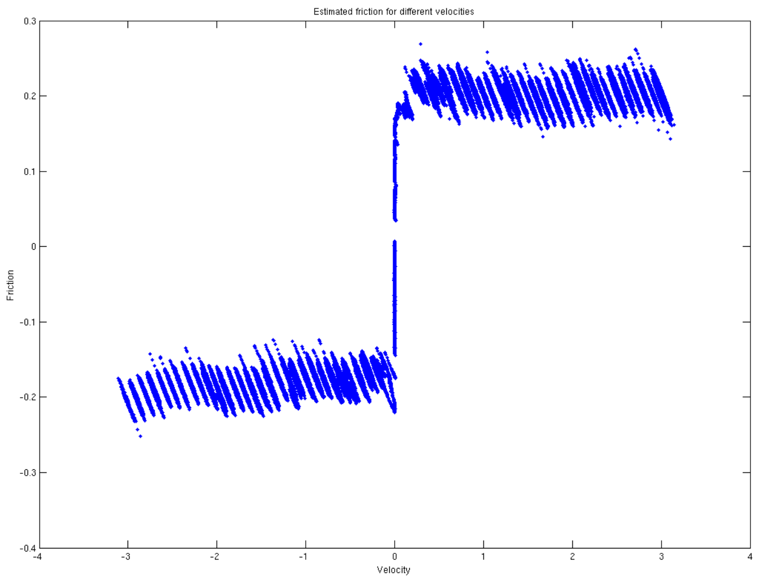


Figure 2: Measured friction of the process for different velocities

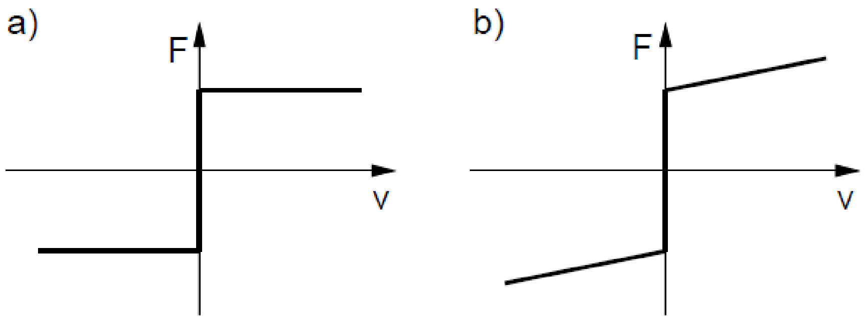


Figure 3: a) Friction model with coulomb friction. b) Friction model with coulomb and viscous friction.

where 6 is model a and only depends on the direction of the velocities and 7 is the model in b where a viscous term that depends on the velocity is added.

The regression models for the two friction models will take the form of 12 and 13 respectively.

$$\phi = sign(\dot{\varphi}), \beta = f_c \tag{8}$$

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

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 4

$$x_{k+1} = A_d x_k + B_d(u_k - f) \tag{10}$$

To get the RLS to work all measurements are taken one step back in time to estimate the parameters from 12 or 13. After rearranging the equation in 10 we get

$$\frac{x_k - A_d x_{k-1} - B_d u_{k-1}}{-B_d} = f \tag{11}$$

where f is one of the regressors in 12 or 13. 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

$$\phi = \begin{bmatrix} sign(\dot{\varphi}) \\ 1 \end{bmatrix}, \beta = \begin{bmatrix} f_c \\ f_o \end{bmatrix} \tag{12}$$

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

5.2 Recursive least squared algotrithm

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

$$\hat{\theta}_k = \hat{\theta}_{k-1} + P_k \phi_k \epsilon_k \tag{14}$$

$$\epsilon_k = y_k - \phi_k^T \hat{\theta}_{k-1} \tag{15}$$

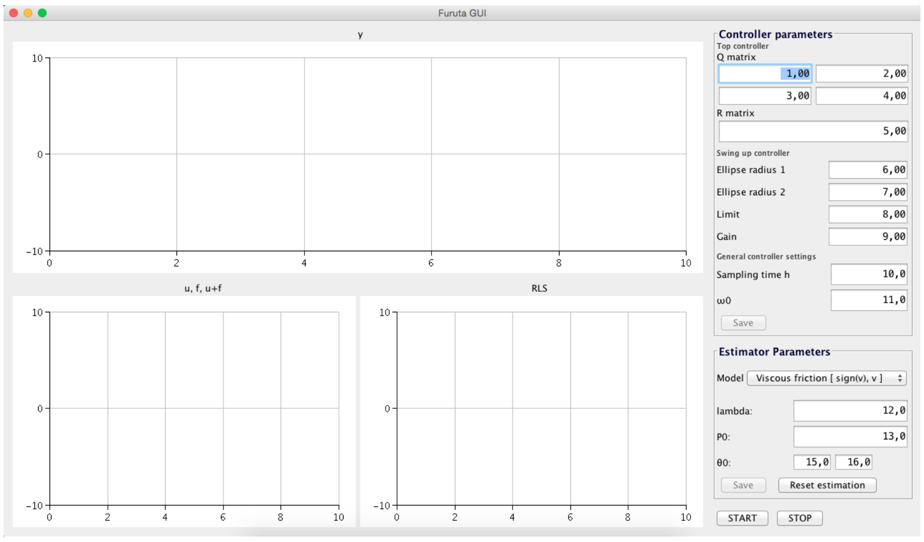
$$P_k = \frac{1}{\lambda}(P_{k-1} - \frac{P_{k-1}\phi_k\phi_k^T P_{k-1}}{\lambda + \phi_k^T P_{k-1}\phi_k}) \tag{16}$$

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 should be able to change controller parameters for both types of controller and for the friction estimator. For the top controller, the Q and R matrices. For the swing up controller, the switching area and its limit and the gain. Other general parameters for the controller such as sampling time h (if possible), and ω_0. And for the friction estimation the forgetting factor λ and the initial values P_0 and ω_0.

The GUI will plot the outputs from the Furuta pendulum, the friction estimation, control signal, both uncompensated as well as the compensated signal. We aim to have different regressor modes for the friction compensation and the operator can therefore change different parameters depending on the regressor chosen.



Controller parameters

Top controller

Q matrix

1,00	2,00
3,00	4,00

R matrix: 5,00

Swing up controller

Ellipse radius 1	6,00
Ellipse radius 2	7,00
Limit	8,00
Gain	9,00

General controller settings

Sampling time h	10,0
ω0	11,0

Estimator Parameters

Model	Viscous friction [sign(v), v]
lambda:	12,0
P0:	13,0
θ0:	15,0 16,0

Figure 4: Overview of the program structure

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 behavior of the arm angle and pendulum angle can be viewed in 5 where 13 was used as a friction model.

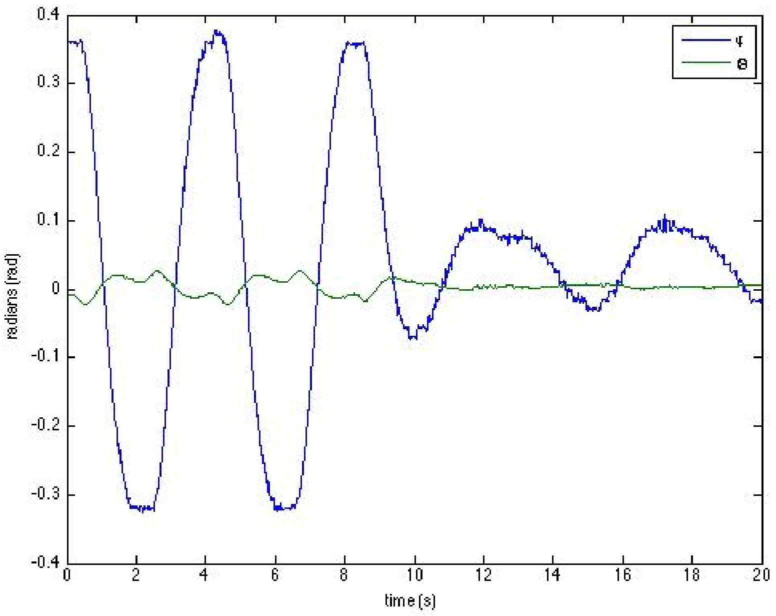
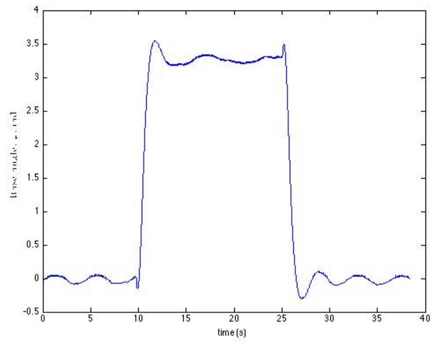


Figure 5: Arm and pendulum angle behavior in steady state and turning on friction compensation after 10 seconds. The parameter estimation, after manual excitation of the system, converged before being turned on.

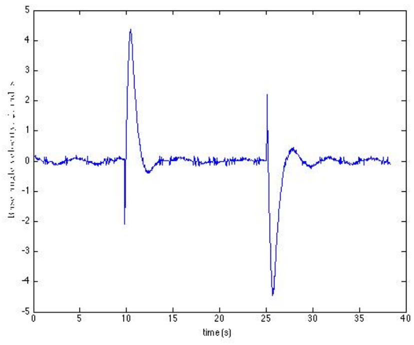
In 5 see that the amplitude of the limit cylces are lowered to about a third of the size. We can also see an offset from the zero angle.

7.2 Coloumb + viscouse friction model

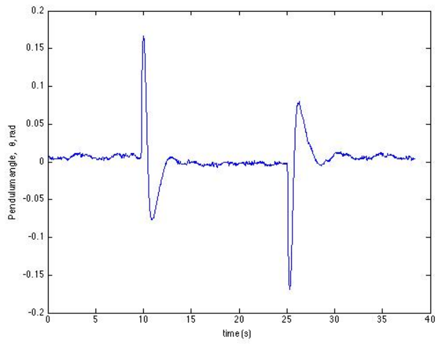
In 6d, **??**, **??**, **??** we can see the behavior of the angle and angular velocity for both the arm and the pendulum.



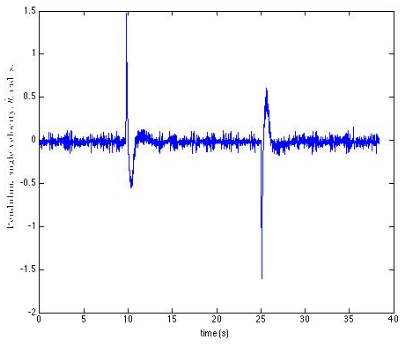
(a) The step response for the angle of the arm.



(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.

8 Conclusion