Integration Project: Reaction Wheel Pendulum

India van Doornen (4959914), Niels Stienen (5595738)
Mechanical Engineering, Delft University of Technology, Delft, The Netherlands

***Abstract*—This report describes the development and validation of controllers for a reaction wheel pendulum system. The setup consists of a pendulum with a mounted rotating flywheel, driven by a DC motor. Two encoders measure the pendulum angle and the flywheel velocity.**

The modelling process begins by splitting the system into two subsystems: the motor-flywheel dynamics and the pendulum dynamics. These subsystems are individually modelled and subsequently combined to form a complete nonlinear model. System identification is performed using MATLAB's nonlinear grey-box estimation tool.

A nonlinear local observer is selected for state estimation due to the accurate fit of the nonlinear model with the actual plant. This observer provides the necessary state estimates for implementing the controllers.

Two controllers, Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC), are designed and tested. Both controllers successfully stabilise the pendulum in its upright, unstable equilibrium, meeting the specified performance criteria. The swing-up control is managed by a finite state machine, effectively transitioning the pendulum from its downward position to the upright equilibrium in five swings.

CONTENTS

India van Doornen and Niels Stienen, are master students at TU Delft, The Netherlands. E-mail addresses: {k.i.vandoornen, n.l.stienen}@student.tudelft.nl.

NOMENCLATURE

Constants

α	Control signal constant
κ_e	Back-EMF constant
κ_t	Motor torque constant
μ_f	Dynamic friction coefficient (flywheel)
μ_p	Dynamic friction coefficient (pendulum)
b_f	Viscous friction coefficient (flywheel)
b_p	Viscous friction coefficient (pendulum)
g	Acceleration due to gravity
i	Current
J_1	Inertia (motor armature)
J_2	Inertia (flywheel)
J_f	Inertia (flywheel + armature)
J_p	Inertia (pendulum)
L	Inductance
l	Distance to centre of mass
m	Combined mass
R	Resistance
$u(t)$	Control input

State variables

$\ddot{\theta}$	Angular pendulum acceleration
$\dot{\omega}$	Angular flywheel acceleration
$\dot{\theta}$	Angular pendulum velocity
ω	Angular flywheel velocity
θ	Angular pendulum displacement

Swing-up symbols

Δ	Stabilising region
$\omega_{\max}$	Maximum angular flywheel velocity
Ψ	Swing-up region
k_p	Slow-down constant
k_u	Swing-up constant
$v_{\max}$	Maximum angular pendulum velocity

I. INTRODUCTION

This report presents the development and validation of controllers for a reaction wheel pendulum system. The primary objective is to stabilise the pendulum in its upright, unstable equilibrium using both Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) strategies. The system employs a nonlinear local observer to estimate the states required for control, leveraging the high accuracy of the developed nonlinear model of the plant. Additionally, a finite

state machine is used to implement the swing-up strategy, which brings the pendulum from its downward position to the upright position where the stabilising controllers take over.

The reaction wheel pendulum setup consists of a rotating flywheel mounted on a pendulum. The flywheel is driven by a DC motor, and the system's dynamics are controlled by varying the motor's input. The torque generated by the flywheel's angular acceleration influences the pendulum's motion, which can be utilized to stabilize it. Two encoders are integrated into the system: one for measuring the angular position of the pendulum with high accuracy and another for monitoring the rotational velocity of the flywheel.

II. Modelling

The reaction wheel pendulum, as seen in Figure 1, is a pendulum that uses the change in momentum of a flywheel as its control input. To derive the equations of motion, the system is first divided into two separate subsystems before combining them. These subsystems consist of the combined motor and flywheel dynamics, and the free-swinging pendulum dynamics.

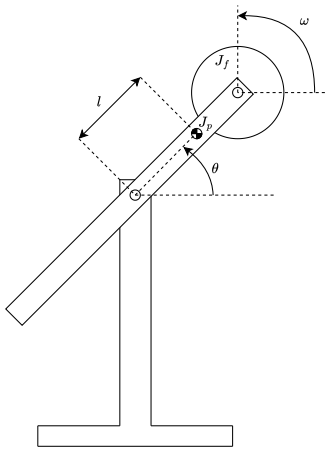


Fig. 1: Schematic drawing of Reaction Wheel Pendulum.

A. Motor and Flywheel

The dynamics of a DC motor are derived from a standard circuit model for a DC motor, illustrated in Figure 2, resulting in a second-order system with current i and flywheel velocity ω as the states.

The change in current depends on the inductance L, resistance R, scaled control input αu, and the back-EMF current resulting from the flywheel velocity $\kappa_e \omega$. The dynamics for the flywheel velocity depend on the combined inertia J_f, viscous and dynamic friction, modelled as $b_f\omega$ and $\mu_f \mathbf{sgn}(\omega)$ respectively, as well as the motor torque described by a constant κ_t times the current. The resulting dynamics are

$$\frac{\mathrm{d}i}{\mathrm{d}t}L + Ri + \kappa_e\omega = \alpha u, \tag{1}$$

$$J_f\dot{\omega} + b_f\omega + \mu_f \mathbf{sgn}(\omega) = \kappa_t i, \tag{2}$$

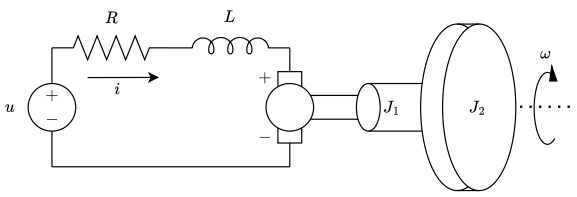


Fig. 2: Schematic drawing of DC motor and flywheel.

with the total inertia J_f defined as the sum of the flywheel inertia J_2 and the motor armature inertia J_1.

Assuming that the inductance constant L is approximately 0, the model can be simplified to a first-order differential equation. Rewriting the above equations, the system dynamics are

$$i = \frac{1}{R}(\alpha u - \kappa_e\omega), \tag{3}$$

$$\dot{\omega} = \frac{1}{J_f}\Big(\frac{\kappa_t\alpha}{R}u - \Big(b_f + \frac{\kappa_t\kappa_e}{R}\Big)\omega - \mu_f \mathbf{sgn}(\omega)\Big). \tag{4}$$

The torque produced by the motor is

$$\tau = J_f\dot{\omega}, \tag{5}$$

$$\tau = \frac{\kappa_t\alpha}{R}u - \Big(b_f + \frac{\kappa_t\kappa_e}{R}\Big)\omega - \mu_f \mathbf{sgn}(\omega). \tag{6}$$

B. Pendulum

The equations of motion for the pendulum are derived by setting the moment of inertia times the angular acceleration equal to the net torque acting on the system,

$$J_p\ddot{\theta} = \tau_g + \tau_b + \tau_\mu \tag{7}$$

where the torques are defined as

$$\tau_g = -mgl\sin\theta, \tag{8}$$

$$\tau_b = -b_p\dot{\theta}, \tag{9}$$

$$\tau_\mu = -\mu_p \mathbf{sgn}(\dot{\theta}). \tag{10}$$

The resulting dynamics of the pendulum are

$$\ddot{\theta} = \frac{1}{J_p}\Big(-mgl\sin\theta - b_p\dot{\theta} - \mu_p \mathbf{sgn}(\dot{\theta})\Big). \tag{11}$$

C. Joint system

To obtain the complete model of the reaction wheel pendulum, the above equations are combined, incorporating the torque produced by the motor as seen in (6), into the pendulum dynamics. The resulting equations of motion are

$$\ddot{\theta} = \frac{1}{J_p}\Big(-mgl\sin\theta - b_p\dot{\theta} - \mu_p \mathbf{sgn}(\dot{\theta}) - \tau\Big), \tag{12}$$

$$\dot{\omega} = \frac{1}{J_f}\Big(\frac{\kappa_t\alpha}{R}u - \Big(b_f + \frac{\kappa_t\kappa_e}{R}\Big)\omega - \mu_f \mathbf{sgn}(\omega)\Big). \tag{13}$$

D. Linearised system

The model is also linearised for use in the design of the control architecture. Since the equations contain nonlinear friction terms dependent on the sign of angular velocities, the model is linearised by first changing all sign terms.

$$\mathbf{sgn}(x) \to x \tag{14}$$

The resulting equations are then linearised around an equilibrium point by taking the Jacobian and substituting in the equilibrium state.

$$A\Big|_{x_{eq}} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{mgl\cos(\theta_{eq})}{J_p} & \frac{-b_p-\mu_p}{J_p} & \frac{-b_f-\frac{\kappa_t\kappa_e}{R}-\mu_f}{J_p} \\ 0 & 0 & \frac{-b_f-\frac{\kappa_t\kappa_e}{R}-\mu_f}{J_f} \end{bmatrix} \tag{15}$$

$$B\Big|_{x_{eq}} = \begin{bmatrix} 0 \\ \frac{-\kappa_t\alpha}{J_pR} \\ \frac{\kappa_t\alpha}{J_fR} \end{bmatrix} \tag{16}$$

III. System Identification

MATLAB's nonlinear grey-box estimation tool is utilised to identify the parameters of the previously defined nonlinear grey-box model. Initially, the system is decomposed into two subsystems: one comprising only the motor and flywheel, and the other representing the pendulum independently of flywheel actuation. After identifying the parameters for these individual subsystems, the models are combined. Subsequently, system identification is performed again on the combined model to improve the accuracy and fit of the model.

Before each test, the pendulum angle encoders are reset around the downward position to ensure reproducibility. Sensor calibration for the encoder was deemed unnecessary, as the measured angle offset over a full rotation was only 0.0029%, which falls well within the sensor's accuracy range. Due to the unavailability of the required equipment, the flywheel encoder was not calibrated. However, any offsets in velocity are addressed during the system identification process and through the feedback mechanisms in the system.

A. Flywheel

To reduce computational demands and enhance model fit, the number of parameters in (4) was reduced, leading to the following simplified Grey-box model

$$\dot{\omega} = K_1 u - K_2\omega - K_3\mathbf{sgn}(\omega), \tag{17}$$

where the constants are defined as

$$K_1 = \frac{\kappa_t\alpha}{J_fR},\ K_2 = \frac{b_f+\kappa_t\kappa_e}{J_fR},\ \text{and}\ K_3 = \frac{\mu_f}{J_f}. \tag{18}$$

For the system identification process, a persistently exciting signal comprising a combination of a sine wave, chirp, and block wave was employed as training data. To isolate the dynamics of the motor and flywheel, the pendulum was secured in a fixed position during the application of this signal. The system identification was initiated with preliminary estimates of K_1, K_2, and K_3. After a single cycle, the model fit improved significantly from -78.93% to 90.2%. Further cycles did not yield additional enhancements in the model. The resultant fit to the exciting signal is depicted in Figure 3.

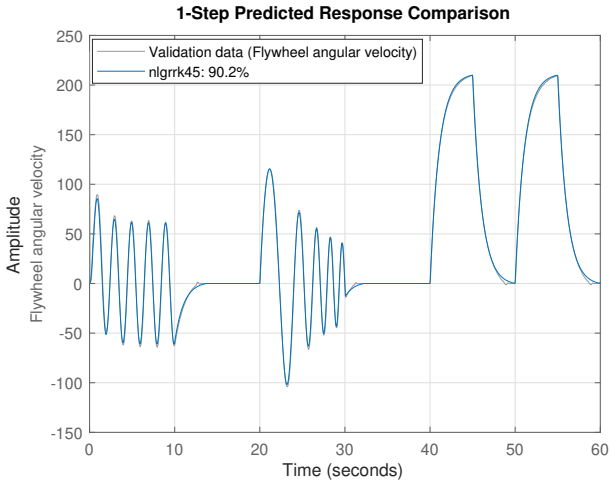


Fig. 3: Model fit on the exciting signal after one cycle of nonlinear grey box model estimation.

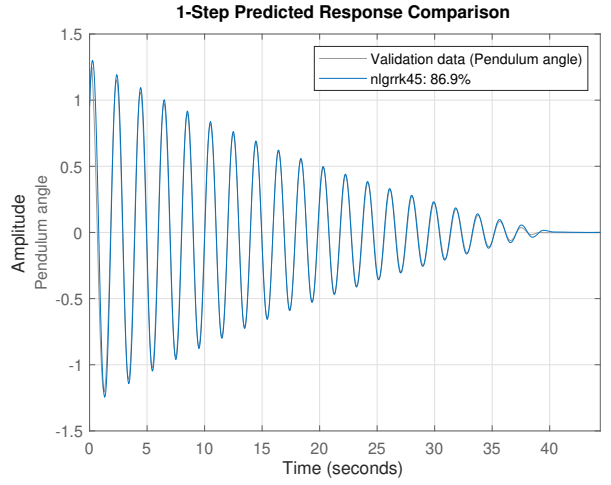


Fig. 4: Model fit on the data of the pendulum damping out from a large angle.

B. Pendulum

The dynamics of the pendulum without flywheel actuation, as described (11), were simplified to

$$\ddot{\theta} = -K_4\sin(\theta) - K_5\dot{\theta} - K_6\mathbf{sgn}(\dot{\theta}), \tag{19}$$

where the constants are defined as

$$K_4 = \frac{mgl}{J_p},\ K_5 = \frac{b_p}{J_p},\ \text{and}\ K_6 = \frac{\mu_p}{J_p}. \tag{20}$$

System identification was conducted for both large and small angles by allowing the pendulum to dampen out without any flywheel actuation. Initial estimates for the constants K_4, K_5, and K_6 resulted in a fit of -91.17% on the large angle data. After two cycles of nonlinear grey-box model estimation, the fit improved significantly to 86.90%, as shown in Figure 4.

The model demonstrates a good fit to the training data. However, it lacks some damping in the latter part of the dataset. This issue is also observed when validating the model

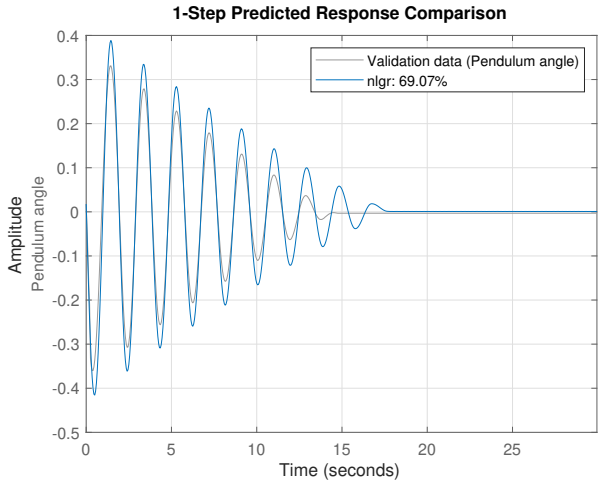


Fig. 5: Model fit on the data of the pendulum damping out from a small angle using the previously identified model.

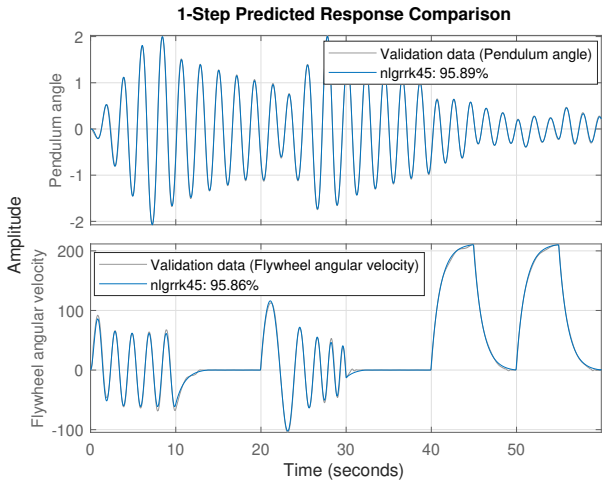


Fig. 6: Model fit on the data of the pendulum being actuated by a persistently exciting signal.

with data from a smaller angle initialisation, as illustrated in Figure 5. Training the model on small-angle data and validating it on large-angle data resulted in a much poorer fit, so it was decided to retain the current model parameters.

C. Total System

Equations (12) and (13) are combined and simplified to

$$\begin{aligned}\ddot{\theta} = &-K_4 \sin(\theta) - K_5\dot{\theta} - K_6\mathbf{sgn}(\dot{\theta}) \\ &- K_\tau(K_1 u - K_2\omega - K_3\mathbf{sgn}(\omega))\end{aligned}, \tag{21}$$

$$\dot{\omega} = K_1 u - K_2\omega - K_3\mathbf{sgn}(\omega), \tag{22}$$

where the constants K_1 to K_6 are defined as previously, and K_τ is defined as

$$K_\tau = \frac{J_f}{J_p}. \tag{23}$$

Nonlinear grey-box model estimation was performed on the combined system, using the values obtained from the decoupled estimations as initial estimates. By allowing all constants to be optimized, the model fit improved significantly from 25.07% for the pendulum angle to 95.89% after one optimization cycle. The fit for the flywheel velocity was similarly impressive at 95.86%, indicating that our nonlinear model closely matches the actual plant dynamics.

D. Validation

To validate the model identified in the previous section, a new dataset was generated using a chirp signal. This signal had an amplitude of 0.7 and varied in frequency from 0.1 Hz to 1 Hz over 30 seconds. The comparison between the measured response and the model's predicted response to the chirp signal is shown in Figure 7. The model demonstrated a fit of 79.02% for the pendulum angle and 91.88% for the flywheel velocity.

It is important to note that this validation was conducted a month after the initial model estimation. During this period, the validation process was primarily carried out within

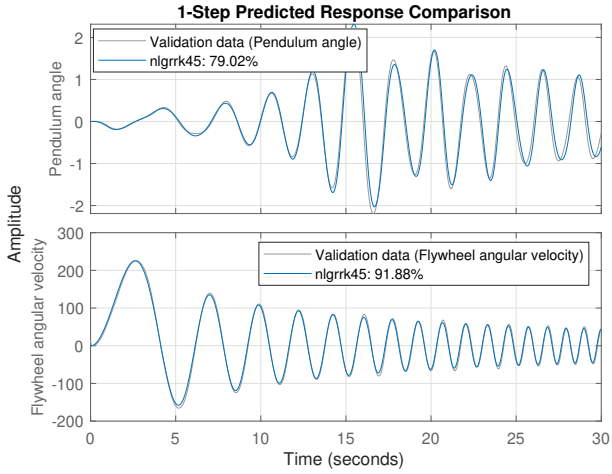


Fig. 7: Validation of the model using a chirp signal: comparison of measured and predicted responses.

Simulink. However, observations indicated that the experimental setup had experienced some degradation over this period, which likely contributed to the relatively lower fit on the validation dataset. Despite this, the model's performance remains well within acceptable limits for the intended application. Consequently, it was determined that re-estimating the model parameters was unnecessary.

IV. Observer Design

Two encoders are used to measure the pendulum angle and flywheel velocity. However, there are no measurements of the pendulum velocity, which means an observer is needed to employ full-state feedback control. Three observer strategies were investigated to reconstruct this state: full-order state observer, reduced-order state observer, and nonlinear local observer. Only the last two will be further elaborated on in this report.

A. Reduced-state observer

Since the measurements of the pendulum angle and flywheel velocity are very accurate and contain little noise, it was argued that it is not necessary to reconstruct these states through a full-order state observer. To reduce computational effort and enhance our understanding of observer strategies, it was decided to instead only reconstruct the pendulum velocity via a reduced-order observer. For this system, the measurements of the other states are directly utilised by the LQR/MPC controller. The derivation of such an observer is presented below.[1]

Consider a system represented in state-space form

$$\dot{x} = Ax + Bu, \tag{24}$$
$$y = Cx, \tag{25}$$

where x denotes the state vector, u is the control input, and y is the measured output.

Given the system's state vector, x can be partitioned as

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{26}$$

where x_1 consists of states that are directly measured and x_2 includes the states that need to be estimated.

The dynamics of the system are then partitioned accordingly

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u, \tag{27}$$
$$y = \begin{bmatrix} C_1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{28}$$

where A_{11}, A_{12}, A_{21}, A_{22}, B_1, B_2, and C_1 are submatrices of A, B, and C partitioned according to the measured and unmeasured states.

A full-order observer for this system is given by

$$\begin{bmatrix} \dot{x}_{1e} \\ \dot{x}_{2e} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_{1e} \\ x_{2e} \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u + \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} (y - C_1 x_{1e}). \tag{29}$$

Given that x_1 is directly measured, x_{1e} can be replaced with $C_1^{-1}y$, where C_1 is invertible by assumption. The resulting observer for the unmeasured states is then

$$\dot{x}_{2e} = A_{21}C_1^{-1}y + A_{22}x_{2e} + B_2u. \tag{30}$$

Let x_{2e} be given by

$$\dot{x}_{2e} = Ly + z, \tag{31}$$

where z is a dummy state with dynamics

$$\dot{z} = Fz + Gy + Hu, \tag{32}$$

for some matrices F, G, and H.

The error dynamics for the system are then defined as $e_2 = x_2 - x_{2e}$, and are consequently grouped and simplified into:

$$\begin{aligned} \dot{e}_2 = Fe_2 &+ (A_{21} - LC_1A_{11} - GC_1 + FLC_1)x_1 \\ &+ (A_{22} - LC_1A_{12} - F)x_2 \\ &+ (B_2 - LC_1B_1 - H)u, \end{aligned} \tag{33}$$

where F, G, and H are parameters that need to be determined such that the error e_2 is driven to zero. This requirement implies the conditions

$$GC_1 = A_{21} - LC_1A_{11} + FLC_1, \tag{34}$$
$$F = A_{22} - LC_1A_{12}, \tag{35}$$
$$H = B_2 - LC_1B_1. \tag{36}$$

Under these conditions, the error dynamics become

$$\dot{e}_2 = Fe_2. \tag{37}$$

If F is chosen such that it is Hurwitz (i.e. $\lambda_i < 0,\ \forall i$) or Schur (i.e. $\rho(F) < 1$), the errors will asymptotically decay to 0. Through pole-placement, the matrix L that ensures this requirement is met can be found. The observer pole is chosen to be faster than the system pole, specifically set at -20 in the continuous-time domain. For the discrete system this results in

$$L = \begin{bmatrix} 18.039 & 0 \end{bmatrix}. \tag{38}$$

B. Nonlinear local observer

Non-linear observers are particularly beneficial in systems where the dynamics are inherently nonlinear and a good model of the non-linear plant is available. Given that the nonlinear model described in Section III had an excellent fit with the actual plant, employing nonlinear observers could offer additional performance on state reconstruction over the previous linear (reduced-order) observer.

The primary goal of a local observer is to ensure that the estimated state $\hat{x}(t)$ converges towards the actual state $x(t)$ of the system. This is expressed mathematically as

$$\tilde{x}(t) := x(t) - \hat{x}(t) \to 0, \quad t \to \infty, \tag{39}$$

where $\tilde{x}(t)$ represents the state estimation error.

The dynamics of the observer are formulated as

$$\dot{\hat{x}} = f(\hat{x}, u) + H[y - h(\hat{x})], \tag{40}$$

where f and h are twice continuously differentiable functions that describe the system dynamics and outputs, respectively. H is the constant observer gain matrix.

To ensure that the estimation error asymptotically drives to zero, it is essential that the error dynamics are stable. In local observers, this is achieved by linearising the nonlinear system around a constant state x_{ss} and a constant input u_{ss}.

$$A = \left[\frac{\partial f(x_{ss}, u_{ss})}{\partial x}\right], \quad C = \left[\frac{\partial h(x_{ss})}{\partial x}\right] \tag{41}$$

Assuming that the pair (A, C) is detectable, an observer gain H can be selected such that the matrix $A - HC$ is Hurwitz. The poles of the observer are placed at $[-20, -25, -60]$ to determine H, resulting in

$$H = \begin{bmatrix} 84.8 & 0 \\ 1497.8 & 0 \\ 0 & 17.3 \end{bmatrix}. \tag{42}$$

V. Control Design

The linearised dynamics form the basis for designing two linear controllers: LQR and MPC. This section outlines the design procedures for both controllers. Given the high accuracy of the previously developed nonlinear model in replicating actual plant behaviour, the controllers were initially tuned at home using plant simulations. This preliminary tuning ensured that the controllers were effectively operational upon implementation in the lab, significantly streamlining the work sessions. Given the high accuracy and signal-to-noise ratio of the encoders, for these state-feedback controllers, the state estimation from the observer is used only for the pendulum velocity. All other states rely on direct measurements.

The controllers were tuned to achieve a rise time of less than 0.5 seconds and a settling time of less than 2 seconds for the flywheel pendulum. The maximum allowable overshoot was set to 5 degrees to prevent excessive oscillations and high control inputs.

A. LQR design

LQR aims to determine the optimal control policy by solving a quadratic cost function that balances the state performance and control effort. The control gain K is derived by minimising this cost function, leading to a feedback law $u_k = -Kx_k$, where K is computed from the solution to the Discrete Algebraic Riccati Equation (DARE).

To prevent abrupt changes in control action, a penalty on the rate of change of the control input was imposed. The resulting cost function is adapted to

$$\sum_{n=1}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k + \Delta u_k^T L \Delta u_k\right). \tag{43}$$

After the introduction of the augmented state $\tilde{x}_k = \begin{bmatrix} x_k & u_{k-1} \end{bmatrix}^T$, the cost function can be reformulated to resemble that of a standard LQR with cross terms

$$\sum_{n=1}^{\infty} \left(\tilde{x}_k^T \tilde{Q} \tilde{x}_k + u_k^T \tilde{R} u_k + 2\tilde{x}_k^T \tilde{M} u_k\right), \tag{44}$$

with

$$\tilde{A} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix},\ \tilde{B} = \begin{bmatrix} B \\ I \end{bmatrix},\ \tilde{Q} = \begin{bmatrix} Q & 0 \\ 0 & L \end{bmatrix},$$
$$\tilde{R} = R + L, \text{ and } \tilde{M} = \begin{bmatrix} 0 \\ -L \end{bmatrix}.$$

The weight matrices determined for achieving the best performance are given by

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.0040 \end{bmatrix},\ R = 5 \times 10^{-4}, \text{ and } L = 20. \tag{45}$$

Additionally, the optimal sampling time for the LQR controller with swing-up functionality was established at $T_s = 0.01$ seconds.

B. MPC design

Model Predictive Control operates by solving an optimisation problem at each time step to determine the optimal control actions over a finite prediction horizon while satisfying state and input constraints. The primary objective is to minimize a cost function defined as

$$V_N(x_0, \mathbf{u}_N) = \sum_{k=0}^{N-1} l(x_k, u_k) + V_f(x_N), \tag{46}$$

where N is the prediction horizon of the controller.

The terminal cost is chosen as a Lyapunov function to ensure (Lyapunov) stability of the closed-loop system. A (polytopic) invariant terminal set can be used to guarantee recursive feasibility of the MPC problem. Alternatively, a significant weight factor $\beta \gg 1$ can be applied to the terminal cost. This strategy heavily prioritises the final state in the optimization process, effectively steering the solution towards a control invariant set without explicitly defining one. Both were not found to be necessary as the finite state machine enforces the MPC to only be active within the terminal set.

The same rate-of-change penalty as for the LQR controller was chosen. The resulting stage and terminal cost are then given by

$$\begin{aligned} l(\tilde{x}_k, u_k) &= \frac{1}{2}\left(\tilde{x}_k^T \tilde{Q} \tilde{x}_k + u_k^T \tilde{R} u_k + 2\tilde{x}_k^T \tilde{M} u_k\right), \\ V_f(\tilde{x}_N) &= \frac{1}{2} \tilde{x}_N^T \tilde{P} \tilde{x}_N, \end{aligned} \tag{47}$$

where P is the unique (positive semi-definite) solution to the DARE using the original dynamics.

$l(\tilde{x}_k, u_k)$ can be expressed as a quadratic function

$$\frac{1}{2} \begin{bmatrix} \tilde{x}_k \\ u_k \end{bmatrix}^T N \begin{bmatrix} \tilde{x}_k \\ u_k \end{bmatrix}, \quad N = \left[\begin{array}{c|c} \tilde{Q} & \tilde{M} \\ \hline \tilde{M}^T & \tilde{R} \end{array}\right] \tag{48}$$

allowing for quadratic programming of the optimization problem. R must be positive definite ($R \succ 0$) and both Q and L must be positive semi-definite ($Q, L \succeq 0$).

For the MPC implementation, the weights of the LQR controller were slightly adjusted, specifically by doubling the rate-of-change penalty weight. The adjusted weight matrices are

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.0040 \end{bmatrix},\ R = 5 \times 10^{-4}, \text{ and } L = 40. \tag{49}$$

A sampling time of $T_s = 0.02$ seconds was found to still give satisfactory results. The prediction horizon was set to $N = 15$. Only an input constraint was imposed, limiting the control between -1 and 1.

C. Swing-up Strategy

The previously designed LQR and MPC are only able to stabilise the pendulum within a limited range near the upright equilibrium. However, to transition the pendulum from the downward position to an appropriate range for stabilisation, a robust swing-up strategy is needed. This report employs a

finite state machine approach inspired by the swinging pirate ship rides found in amusement parks.

In the vicinity of the downward equilibrium, the control strategy actively accelerates the pendulum based on the direction of its velocity. When the pendulum exits this specified range, the control effort is disengaged by setting the input to zero. This cycle allows the pendulum to progressively increase its amplitude through successive "Swing-up" and "Nothing" states until it enters the "Stabilising" range. Here, the LQR or MPC can effectively take over control. This swing-up strategy is visually illustrated in Figure 8.

To ensure the pendulum does not exceed angular velocities that the controllers can handle, a "Slow Down" state is integrated. This state activates when the pendulum's angular velocity becomes excessively high within the "Nothing" state, effectively reducing its speed to manageable levels. Additionally, as a safety measure, the flywheel's angular velocity is capped at 250 rad/s within the "Stabilising" region through a transition to the "Speed limit" state. This architecture is visually represented in Figure 9, depicting the transitions and operational states of the finite state machine.

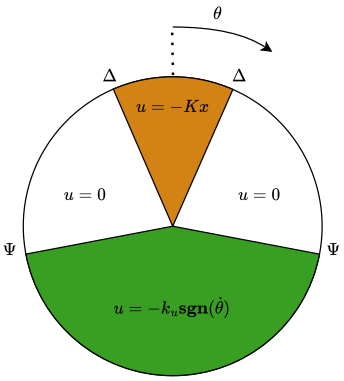


Fig. 8: Overview of the swing-up strategy regions without constraints on flywheel velocity or pendulum velocity for the LQR controller. Δ and Ψ represent the angles from which the control strategy changes.

VI. Results

This section covers the results of the designed nonlinear local observer, the stabilising performance for both the LQR and MPC controller, and the swing-up control results. The corresponding code and datasets are available in the GitHub repository. [2].

A. Observer validation

For conciseness, this report includes validation only for the nonlinear local observer, as it was selected over the reduced-order observer.

To demonstrate that the local observer quickly converges to the actual state, the measured and estimated states are compared during the swing-up phase, as shown in Figure 10.

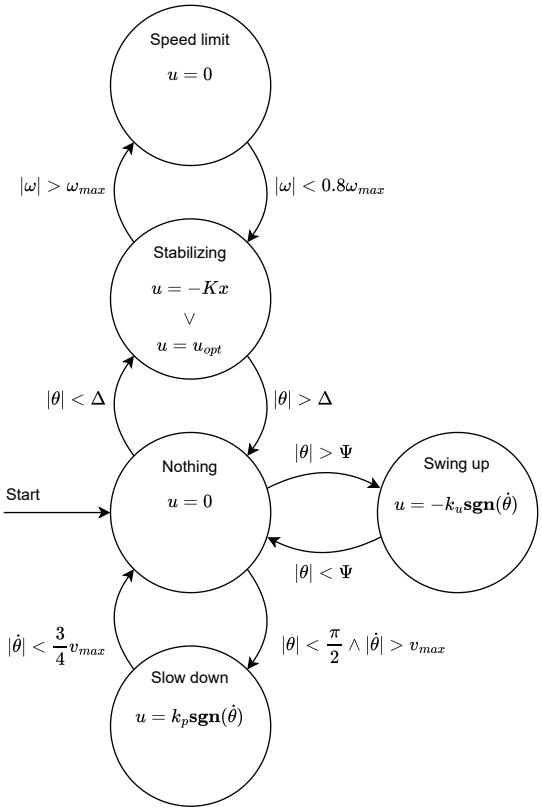


Fig. 9: Finite state machine implementation of the swing-up strategy. Angles Δ and Ψ are the same as in Figure 8.

The results indicate that the nonlinear local observer tracks the measured states accurately, confirming its effective design. Since the pendulum velocity estimate cannot be directly compared to a measurement, it is instead compared to the derivative of the pendulum angle, presented in Figure 11. The alignment between the estimated and calculated pendulum velocities is good, though the observer shows some sensitivity to variations in the pendulum angle. Adjusting the pendulum velocity pole speed could reduce this sensitivity, but it was found to negatively impact controller performance. The chosen pole speed achieves a balanced trade-off between sensitivity to sensor noise and control performance.

B. Stabilising control

The performance of both the LQR and MPC controllers was first tested for stabilising the pendulum from a deviation of approximately 10-15 degrees from the unstable equilibrium. Figure 12 demonstrates that both controllers successfully stabilised the pendulum, achieving a rise time of around 0.4 seconds, a settling time of around 1.5 seconds, and an overshoot of only 2 degrees. These results meet the specified controller requirements.

Next, the pendulum was disturbed during operation by tapping it away from the equilibrium with increasing force,

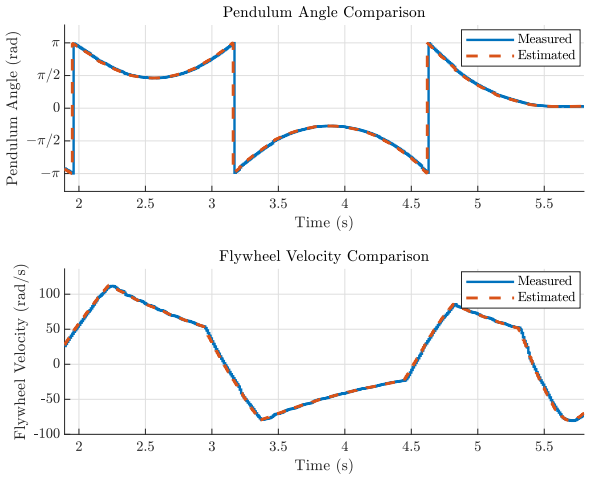


Fig. 10: Comparison of measured and estimated states during the swing-up phase. The angles are wrapped between $-\pi$ and π, explaining the sudden changes in pendulum angle.

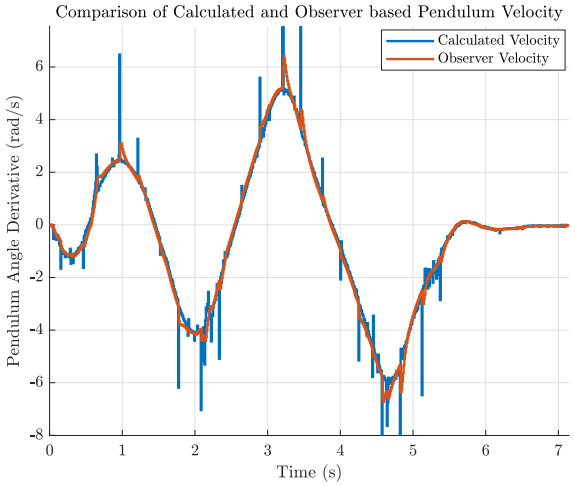
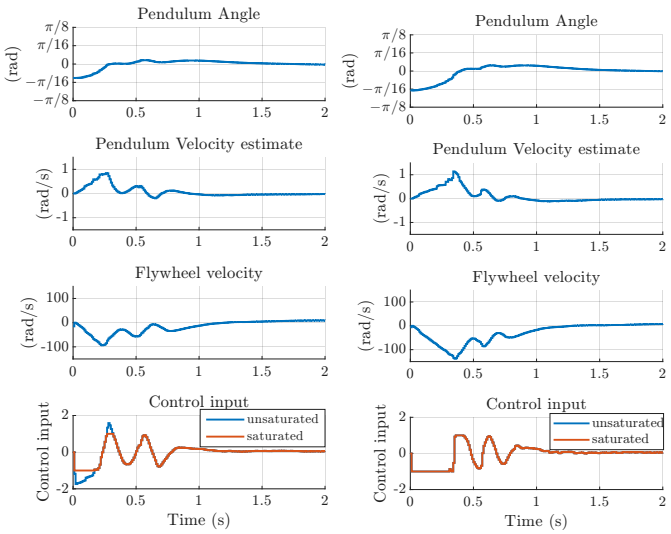


Fig. 11: Comparison of estimated pendulum velocity with the derivative of the pendulum angle.

as shown in Figure 13 and Figure 14. Both controllers were able to quickly re-stabilise the system, with a disturbance of 20 degrees appearing to be the maximum deviation the system can handle. This limitation arises from the maximum allowable control input not being able to generate a sufficient change in momentum quickly enough, setting an upper limit on the range of θ for which the system is stabilised.

Notably, a strong perturbation was applied to the LQR controller at the 12-second mark in Figure 13, causing the controller to briefly transition to the "Speed limit" state. This is evidenced by the flywheel velocity reaching the 250 rad/s limit, resulting in the control input being momentarily set to zero, allowing the flywheel velocity to decrease. This mechanism effectively prevented system limits from being exceeded while still allowing the pendulum to stabilise.



(a) Stabilising in the unstable equilibrium with LQR.

(b) Stabilising in the unstable equilibrium with MPC.

Fig. 12: Comparison of stabilisation between LQR and MPC.

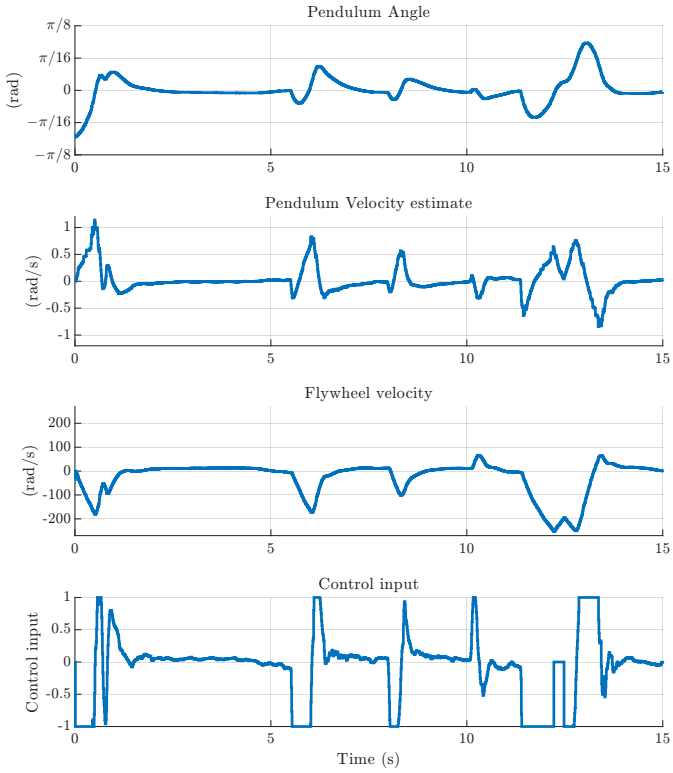


Fig. 13: Stabilising in the unstable equilibrium with LQR under induced disturbances.

C. Swing-up Control

For swing-up control, both systems utilise the finite state machine with the stabilising input being either $u = -Kx$ for LQR or $u = u_{opt}$ for MPC. The system swings the pendulum up into the unstable equilibrium across five swings, starting from a standstill in the stable equilibrium. Once the pendulum approaches the unstable equilibrium, the stabilising control action takes over, successfully stabilising the system with no

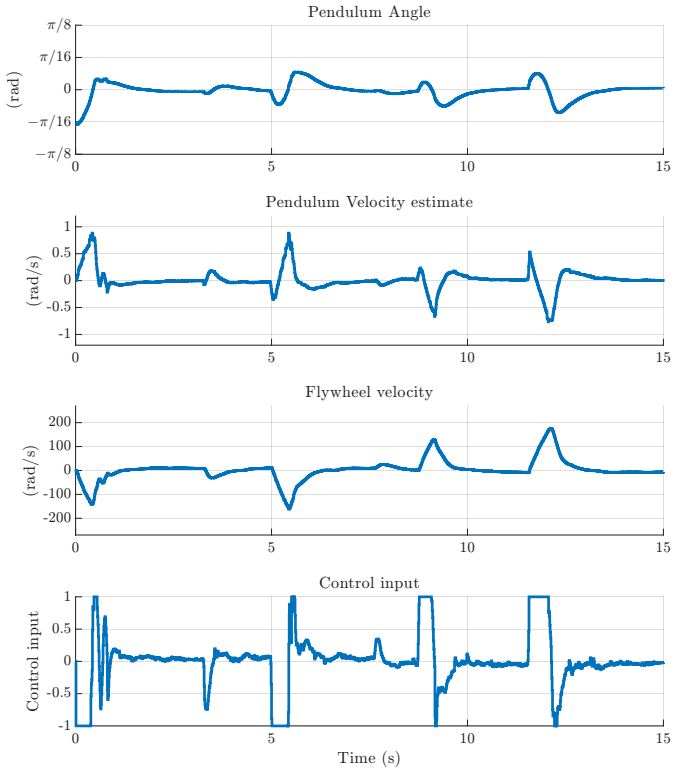
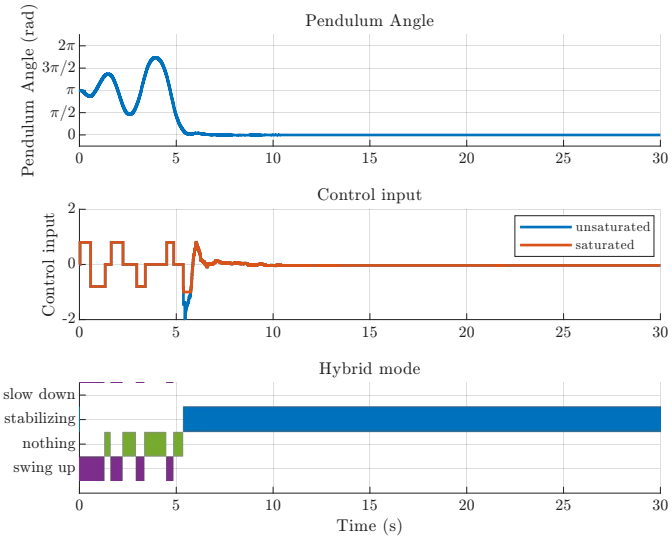
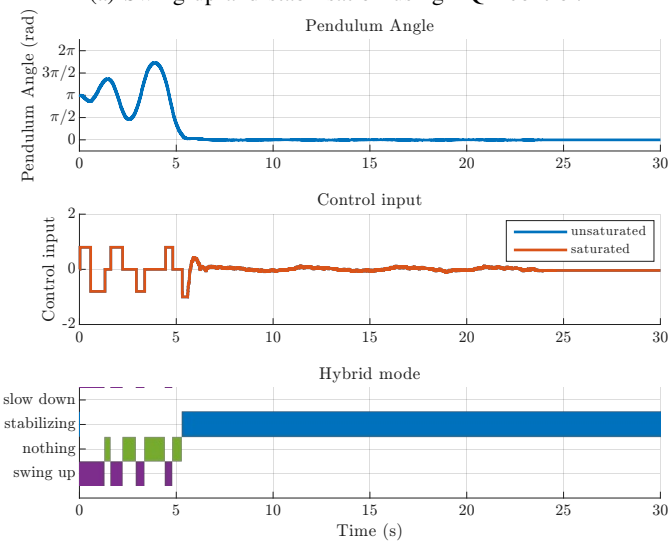


Fig. 14: Stabilising in the unstable equilibrium with MPC under induced disturbances.



(a) Swing-up and stabilisation using LQR control.



(b) Swing-up and stabilisation using MPC control.

Fig. 15: Comparison of swing-up and stabilisation between LQR and MPC controllers.

overshoot, indicating that the pendulum was not moving too fast.

After some time, both controllers manage to position the pendulum such that the static friction in the system is significant enough to keep it in place for the remainder of the run without requiring additional control input. This phenomenon relies on friction to a degree not accounted for in either the linear or nonlinear dynamics models on which the controllers are based, and thus it does not occur consistently. From our experience, this static friction stabilisation happens in approximately one-third of the swing-up simulations from a standstill.

VII. Discussion

The LQR and MPC controllers, combined with a nonlinear local observer, have successfully stabilised the flywheel pendulum in its unstable (upright) equilibrium, meeting the design specifications. The implementation of a finite state machine effectively swings the pendulum up, allowing the controller to take over within just five swings.

Looking back, there are a few areas where the project could have been improved. More time could have been spent on accurately deriving the equations of motion. This would have simplified the system identification process, as there was an error initially present in the equations that was only corrected when writing the report. Although this issue was resolved by substituting the parameters K_1 through K_7, starting with correct equations would have streamlined the process. Additionally, a more structured approach to validating intermediate steps, rather than focusing solely on getting the entire system operational, would have been beneficial. This could have helped identify and address issues earlier, leading to a more efficient development process.

Further improvements could focus on implementing an energy-based swing-up strategy instead of the current finite-state machine implementation. This approach would adapt the control input based on the system's current energy, potentially reducing the number of swings needed to reach equilibrium. Alternatively, integrating the swing-up process into the MPC controller through hybrid architectures could enhance performance.

Currently, there is no algorithm in place to ensure a bumpless control transfer when switching from the swing-up to the stabilizing controller. Implementing such a strategy would ensure a smooth transition between controllers, preventing sudden changes in the control action and resulting in a

smoother swing-up process.

Additional effort could be dedicated to reliably stabilising the pendulum to a standstill. This might involve refining the system identification, fine-tuning the controllers, or employing gain scheduling based on the pendulum's angle and velocity.

Although the nonlinear local observer has performed adequately in reconstructing the pendulum velocity state for state feedback control, its sensitivity to sensor deviations or noise could be improved. Further tuning or exploring alternative state reconstruction strategies, such as Extended Kalman Filters, could enhance state estimation, thereby improving the overall performance of the controllers.

REFERENCES

[1] J. Glower, "Lecture 22: Reduced-order observers," Lecture presented at NDSU, ECE 463/663, 2023, available from Bison Academy. Lecture notes, homework sets, and solutions.

[2] K. I. van Doornen and N. Stienen, "Swing-up and stabilization of a reaction wheel pendulum," 2024. [Online]. Available: https://github.com/StienenNiels/IntegrationProject

APPENDIX A
SIMULINK BLOCK DIAGRAM

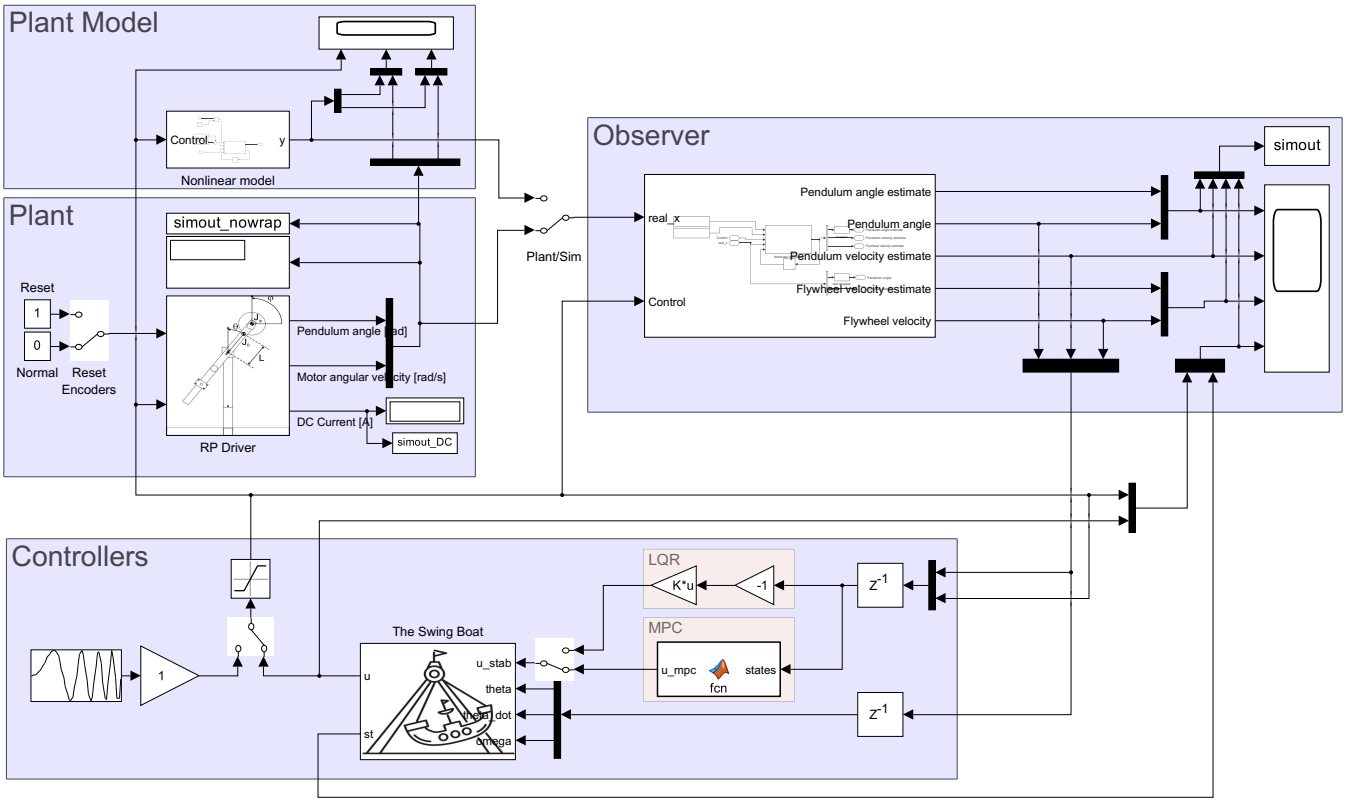


Fig. 16: Simulink model of the plant with observer and controllers. The top left section includes both the real plant and the nonlinear model, labelled as ”Plant” and ”Plant Model” respectively. Depending on the availability of the setup, one of these models is fed into the Observer, which then provides a full state estimate to the controller region. The controller region comprises the LQR and MPC controllers, which feed into the finite state machine responsible for swing-up control.