FRTN55 Automatic Control, Advanced Course

Laboratory Session 2

Kalman Filtering and LQ Control of the MinSeg Robot[1]

Department of Automatic Control
Lund University

1. Introduction

This laboratory session teaches the student how to develop *Kalman filters* and *linear-quadratic* (LQ) controllers. The control structure is developed and applied to the MinSeg™ balancing robot, see Figure 1.

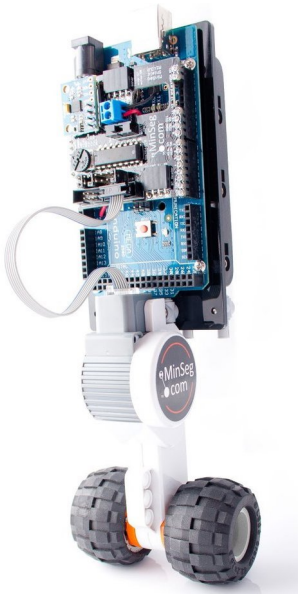

Figure 1 The MinSeg™ balancing robot.

Main Concepts

The aim of the lab is to develop a working controller for balancing the robot and, optionally, let it follow a square-wave reference signal for the wheel position. First, two Kalman filters will be designed to extract state information from the raw gyro, accelerometer, and wheel encoder signals. Thereafter, an LQ controller will be designed for state feedback control from the estimated states with optional integral action and reference tracking.

Pre-lab assignments

- Read this document,
- Complete all assignments marked as ***Preparatory***,
- Bring the code used to solve ***Preparatory*** exercises as a script to the lab session **(use the provided code skeleton)**,
- Have a good grasp of the material from Lectures 5–9 on sampled-data control, optimal state feedback (LQR), Kalman filtering, and LQG control.
- Go through Exercise 8.

[1] Written by Anton Cervin and Nils Vreman, latest update September 12, 2023.

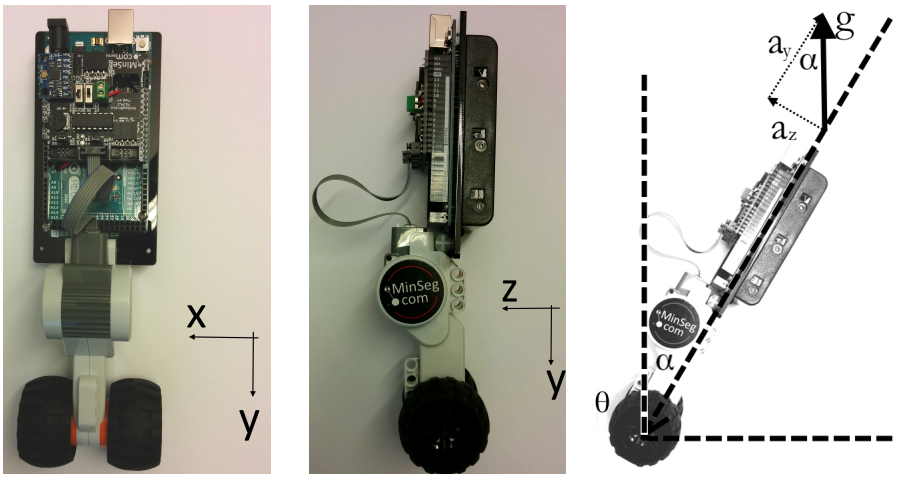

Figure 2 Definition of x, y, and z axes, tilt angle α and wheel angle θ (adapted from [1]).

2. The Process

The drivetrain of the MinSeg (model M1V4) is a Lego NXT DC motor equipped with wheels. An Arduino Mega 2560 microcontroller drives the motor and reads sensor data. During the lab the MinSeg will be connected to a PC via a USB connector. This allows the Arduino to be programmed, plot data to be read, and parameters to be updated during runtime.

The MinSeg is equipped with two sensor units. The first is a rotational encoder, built into the Lego NXT motor, which gives the wheels' rotational position around the wheel axle; we will call this angle θ. In the conditions of the lab, no wheel slippage will occur, so this angle the directly corresponds to backward and forward position of the robot.

The second sensor unit is the *inertial measurement unit* (IMU). It is part of the board on top of the Arduino Mega. An IMU consists of two sensors: a gyro and an accelerometer. A gyro gives the angular velocities around its coordinate axes, while the accelerometer gives the acceleration along the axes. Figure 2 shows the x, y and z axes of the IMU's coordinate system. Figure 2 also shows the robot's tilt angle, α.

In order to properly control the process, an accurate estimate of its state is needed. Without going into details of the dynamics of the system, we say that the state consists of four state variables:

- α: the tilt angle,
- $\dot{\alpha}$: the time derivative of α,
- θ: the angle of rotation of the wheels,
- $\dot{\theta}$: the time derivative of θ.

Since we only measure three of these states directly (θ, α, and $\dot{\alpha}$) and those measurements are very noisy, a state estimate needs to be formed by filtering of the data provided from the IMU and the wheel encoder.

3. The Lab Interface

We will use Simulink with Simulink Coder for modeling and implementation of the filters and controllers. The main diagram is shown in Figure 3.

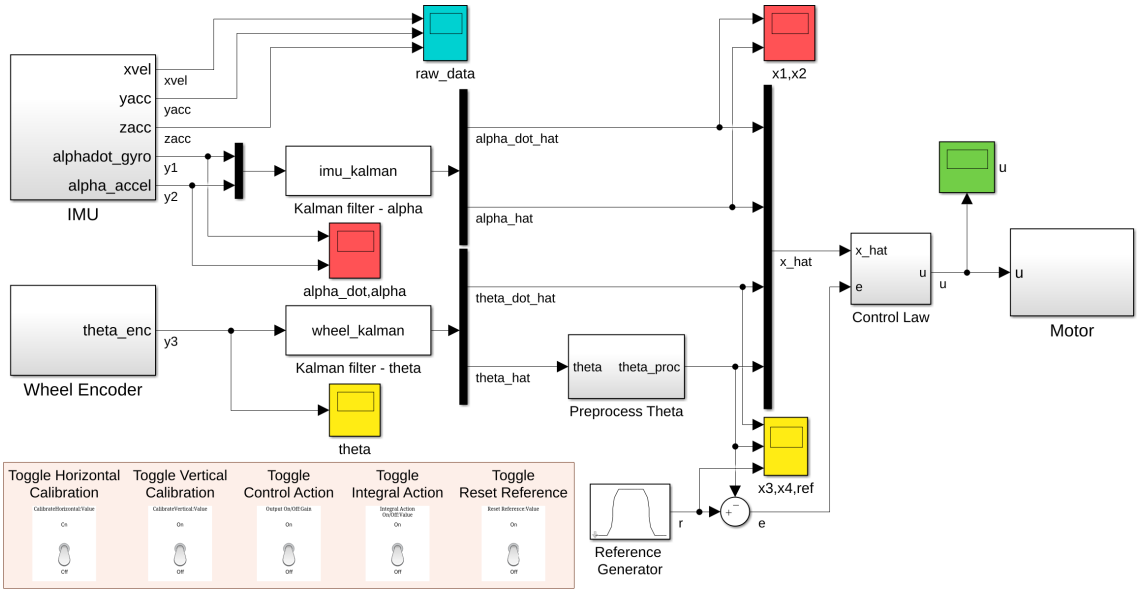

Figure 3 Simulink model `lab2.slx` for filter and controller implementation.

The model is configured by five variables in the Matlab workspace; these variables are `imu_kalman`, `wheel_kalman`, `feedback_gain`, `integral_gain`, and `Ts`. When the Simulink model is run, it will read these variables, compile the resulting controller and upload it to the MinSeg. To design our controller we simply assign different Kalman filters and feedback gains to the workspace variables. When running the setup script, default values (zero) for the controller variables are automatically defined.

The last variable, `Ts`, is the sample time for the controller. We will be working with a sample time of 10 ms. This value is a trade-off between control performance and plotting speed.

Control Panel

At the bottom of Figure 3 you will see a control panel containing 5 switches. These switches exist for your convenience and their behaviour is briefly described below.

- **Toggle Horizontal Calibration:** When the MinSeg is lying in a horizontal position (battery case against the table) flick this switch to automatically calibrate the x and y component of the accelerometer. Calibration takes about 5-10 seconds. When horizontal calibration is finished, turn it off and calibrate the vertical position.
- **Toggle Vertical Calibration:** Place the MinSeg upright (lean it against a coffee mug or the computer screen) and flick this switch to automatically calibrate the z component of the accelerometer. This is extremely delicate; *try to balance it as close to its upright equilibrium as possible.* Calibration takes about 5-10 seconds. When vertical calibration is finished, turn it off; calibration is now finished.
- **Toggle Control Action:** Toggle this switch to turn the motor actuator on and off.
- **Toggle Integral Action:** Toggle this switch to turn integral action on and off.
- **Toggle Reset Reference:** Toggle this switch whilst control action and integral action is off to reset the reference value and integrator.

Note: If you are unable to see the switches in the simulink model, zoom in (they are quite small).

Assignment 1. Login with your student account. Download `lab2_files.zip` from the course homepage and extract the contents to some suitable working directory. Open a terminal window by pressing the *Windows Key* on your keyboard and search for "Terminal". In the terminal window, type

```
VERSION=R2016a matlab
```

to start Matlab R2016a. Once Matlab has started, go into the `lab2_files` directory and then type

```
setup_lab2
```

to setup the paths to the Matlab/Simulink support packages and the Simulink libraries for the Arduino and MinSeg hardware. **Note:** It is important that you run this command directly after starting Matlab.

□

4. Tilt Angle Measurements

First, we will focus on the measurements that will form our estimation of the tilt angle, α, and its derivative, $\dot{\alpha}$. These measurements will be taken from the IMU. The gyro gives us a noisy signal proportional to $\dot{\alpha}$, while the accelerometer data together with some trigonometry can give a rough estimate of α.

4.1 Calibration

The IMU needs to be calibrated. Both the gyro and the accelerometer have an offset that needs to be corrected by adding a bias to the raw signal. This offset can also drift slightly so the IMU might need recalibration during the lab. This calibration has been automated for your convenience, i.e., the offset is automatically removed when you toggle the calibration switches. However, the automatic calibration still needs to be started manually.

Assignment 2. Connect the MinSeg to the computer using the USB cable if it is not already connected. Open the Simulink model `lab2.slx`.

Press `Run` in the Simulink model and wait about 60 seconds for the diagram to be compiled and uploaded to the Arduino (if you get an error message, ask the lab supervisor for help).
Note: it will take about 60 seconds every time you compile and upload the model to the Arduino. Therefore, try to minimize the number of times this has to be done.

When the model is running, open up the raw_data scope (coloured blue) and study the raw signals from the x gyro and from the z and y accelerometers. Rotate the robot in different directions by hand and verify that the signals seem to behave as expected. □

Assignment 3. Lay the robot flat on its back (battery case towards the table) and calibrate the x gyro and y accelerometer readings by toggling the Toggle Horizontal Calibration switch. Wait approximately 5-10 seconds before flicking the switch back again. Then, balance the robot in the upright position (preferable against something heavy, e.g., a coffee cup) and calibrate the z accelerometer reading by flicking the Toggle Vertical Calibration switch; again, wait approximately 5-10 seconds before turning the calibration off. Due to the sensitivity of the sensors, it is *highly recommended* to let the table and MinSeg be perfectly still during calibration. The closer to the upright equilibrium point you manage to calibrate the z accelerometer, the better your controller will perform. □

NOTE: The MinSeg needs to be recalibrated every time it is restarted.

4.2 Angle Measurement

The gyro gives us a direct, but noisy, measurement of the angular velocity, i.e. $\dot{\alpha}$, but we have no direct measurement of α. A rough estimate can be formed by looking at the components of the data given by the accelerometer.

When the device is sitting still, the three acceleration components a_x, a_y, and a_z will add up as

$$\sqrt{a_x^2 + a_y^2 + a_z^2} = 9.81 \text{ m/s}^2$$

As long as the robot is stationary and not tilting sideways ($a_x = 0$), we can use the geometric relationship indicated in Figure 2 and calculate the tilt angle according to

$$\alpha = \mathsf{atan2}(a_z, -a_y)$$

The accelerometer signals are noisy and also pick up any external forces acting on the IMU chip (remember $F = m \cdot a$), making the calculation above meaningful only for low-frequency signal components (below, say, 1 rad/s).

4.3 Measurement Noise Identification

The IMU block in the Simulink model returns the calibrated and rescaled values of α and $\dot{\alpha}$ and these are now considered as two of our noisy measurements. In order to design good filters we would like to know more about the noise characteristics.

Assignment 4. Keep the robot completely still for 60 seconds and then press `Stop`. This stores the 1000 most recent measurements in the workspace under the variable names alphadot_gyro and alpha_accel. For the first measurement (alphadot_gyro), plot the signals, remove any linear trends, calculate the variance, and save it as `y1_var`. Thereafter, the signals spectrum can be plotted. The following code snippet will help you:

```
plot(alphadot_gyro)                    % plot
y1 = detrend(alphadot_gyro,'linear'); % remove linear trend
plot(y1)                               % plot again
pwelch(y1)                             % plot periodogram (estimate of spectrum)
y1_var = var(y1);                      % calculate stationary variance
```

Make sure that you work in a Matlab script for this (as well as all other assignments) such that everything can be repeated easily.

How white is the measurement noise? What is the stationary variance? Answer the same questions for the measurement given by alpha_accel – store the variance of alpha_accel as `y2_var`. Make sure to record the values of `y1_var` and `y2_var` for later use. □

5. Tilt Angle Estimation

We will now use a Kalman filter to reduce the effect of the noise on our measurements y_1 and y_2 of $\dot{\alpha}$ and α. To be able to develop a Kalman filter, a model of the dynamics is needed. The simplest possible choice is to simply consider the robot dynamics as completely unknown and describe them using some process noise in the form of an external angular acceleration w_α. The system can then be modeled as a double integrator from w_α to the tilt angle α, see Figure 4.

The noise parameter w_α contains all dynamics from the motor and the inverted pendulum and is in reality non-white noise. However, since the aim is simplicity we will assume it is white noise with intensity R_1. The downside of this modeling choice is of course that the resulting filter may not be as good as it could be, but it will be adequate for this application.

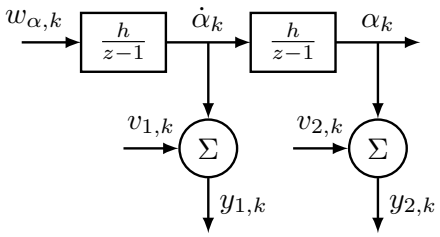

Figure 4 Model of the IMU for design of the first Kalman filter.

Assignment 5 (Preparatory). Convert the model in Figure 4 to discrete state-space form using the state vector $\left(\begin{smallmatrix} x_{1,k} \\ x_{2,k} \end{smallmatrix}\right) = \left(\begin{smallmatrix} \dot{\alpha}_k \\ \alpha_k \end{smallmatrix}\right)$ and the unknown sample time h. Assume that the process is only driven by noise, i.e., $\Gamma = D = 0$. In other words, find Φ, G, and C such that the system can be written as:

$$\begin{cases} x_{k+1} & = \Phi\, x_k + G\, w_{\alpha,k} \\ y_k & = C\, x_k + v_k \end{cases}$$

What *dimensions* and *structure* do the process and measurement noise intensity matrices R_1, R_2, and R_{12} have? Assume that the noise processes are uncorrelated. By dimensions we mean the matrix sizes. Similarly, for the structure of the noise intensity matrices we ask for what will appear in each matrix's individual elements.

Using `dlqe` in Matlab, calculate the discrete Kalman filter gain L_{imu} and the resulting observer poles for some different values of R_1 (very large and very small). *Assume that R_2 is an identity matrix of appropriate size for the* ***Preparatory*** *exercise.*

```
[L_imu, ~, ~, Eig_imu] = dlqe(Phi_imu, G_imu, C_imu, R1, R2);
```

How is the relative magnitude between R_1 and R_2 influencing the speed of the observer? Assume that the sample time is 10 ms. □

Assignment 6 (Preparatory, and at the lab). Design the discrete-time Kalman filter for $\dot{\alpha}$ and α in Matlab, using R_2 from Assignment 4 (or $R_2 = \left(\begin{smallmatrix} 2\cdot 10^{-6} & 0 \\ 0 & 10^{-5} \end{smallmatrix}\right)$ for the ***Preparatory*** exercises), the knowledge you have about the noise intensity matrix from the preparatory exercise (Assignment 5), and some arbitrary value for R_1 as a starting point.

Formulate the Kalman filter as a discrete state-space system, named `imu_kalman`, according to

$$\begin{cases} \hat{x}_{k+1|k} & = \Phi_{imu}\left(I - L_{imu}C_{imu}\right)\hat{x}_{k|k-1} + \Phi_{imu}L_{imu}y_k \\ \hat{x}_{k|k} & = \left(I - L_{imu}C_{imu}\right)\hat{x}_{k|k-1} + L_{imu}y_k \end{cases}$$

Using `ss` in Matlab, we get the following function call

```
imu_kalman = ss( ...                          % The state-space representation
    Phi_imu*(eye(2) - L_imu*C_imu), ...       % The "Phi" matrix
    Phi_imu*L_imu, ...                        % The "Gamma" matrix
    eye(2) - L_imu*C_imu, ...                 % The "C" matrix
    L_imu, ...                                % The "D" matrix
    Ts);                                      % The sample time
```

If your model is correct, the Kalman filter should have two inputs and two outputs.

Plot the Bode magnitude diagram of the filter using `bodemag` and interpret what you see. How are the measurements y_1 and y_2 combined to produce the estimates $\hat{x}_1$ and $\hat{x}_2$ respectively? For balancing, the filter bandwidth from y_1 to $\hat{x}_1$ should be *approximately* 50 rad/s and from y_2 to $\hat{x}_2$ *approximately* 0.5 rad/s. We have provided a function `bandwidth_mimo.m` which calculates the bandwidth for a MIMO system as □

```
bw_mimo = bandwidth_mimo(imu_kalman)
```

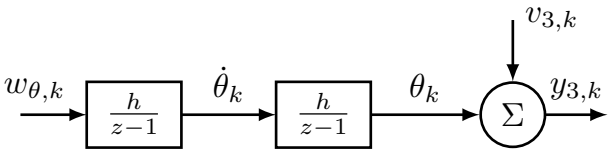

Figure 5 Model of the wheels for design of the second Kalman filter.

6. Wheel Position Estimation

With a filter for two of our state variables, we turn to designing a second Kalman filter for estimating the wheel angular position, θ, and speed, $\dot{\theta}$. We will utilize the rotational encoder of the motor which gives us the last measurement, y_3.

Similar to before, we will model most of the dynamics as unknown process noise on a double integrator. Furthermore, the noise contains the motor's response to changes in the applied voltage and inertia of the robot. A model of the subsystem is shown in Figure 5.

Unlike earlier, only one measurement is now available – the output from the encoder. The signal is not very noisy, however quantized with a resolution of 0.5 degrees, resulting in inaccuracies. The measurement noise of the model, v_3, represents the quantization error of the wheel encoder.

Assignment 7 (Preparatory). Convert the model in Figure 5 to state-space form using the state vector $\left(\begin{smallmatrix} x_{3,k} \\ x_{4,k} \end{smallmatrix}\right) = \left(\begin{smallmatrix} \dot{\theta}_k \\ \theta_k \end{smallmatrix}\right)$.

What *dimensions* and *structure* do the process and measurement noise intensity matrices R_1, R_2, and R_{12} have in this case? Assume that the noise processes are uncorrelated. By dimensions we mean the matrix sizes. Similarly, for the structure of the noise intensity matrices we ask for what will appear in each matrix's individual matrix elements. □

Assignment 8 (Preparatory). Design the Kalman filter for $\dot{\theta}$ and θ in Matlab. Aim for a filter bandwidth of approximately 50 rad/s from $y_3 \to \hat{x}_4$. *Assume that R_2 is the identity matrix of appropriate size.* Formulate the discrete Kalman filter as a state-space system `wheel_kalman` using `ss` and analyse its Bode plot (similarly to Assignment 6). If your model is correct, the Kalman filter should have one input and two outputs. □

Assignment 9. Run the Simulink model and try both the Kalman filters on the real process. Tilt the robot by hand and observe how fast the estimates $\hat{x}_1$ and $\hat{x}_2$ are following the movements.

Turn the robot wheels by hand and verify that the estimates $\hat{x}_3$ and $\hat{x}_4$ seem to behave as expected.

Note that if you need to change your Kalman filters, you can change the covariance matrices in your script, run it, and then update the model parameters in Simulink by clicking anywhere in the Simulink model and pressing `Ctrl+D`. □

7. Design of LQ State Feedback

With filters for all of our state variables we now turn to modeling and controlling the dynamics of the robot to make it balance in the upright position ($\alpha = 0$).

First-principles modeling of the motor, wheels and body of the robot gives a set of nonlinear differential equations, see [2] for details. Linearization of these equations around the upright equilibrium gives the following linear model:

$$\ddot{\alpha} = -3.1\dot{\alpha} + 58.4\alpha + 62.7\dot{\theta} - 148u$$
$$\ddot{\theta} = 40.1\dot{\alpha} - 318\alpha - 766\dot{\theta} + 1808u$$

The control signal u represents the motor voltage (limited to ± 3.25 V for power over USB). Using the state vector $x = [\,\dot{\alpha} \quad \alpha \quad \dot{\theta} \quad \theta\,]^T$ we can write this as

$$\dot{x} = \begin{pmatrix} -3.1 & 58.4 & 62.7 & 0 \\ 1 & 0 & 0 & 0 \\ 40.1 & -318 & -766 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} x + \begin{pmatrix} -148 \\ 0 \\ 1808 \\ 0 \end{pmatrix} u$$

Assignment 10 (Preparatory). Using Matlab, discretize the linear system (with sample time 10 ms) and calculate the poles. The fastest pole is related to the motor dynamics. Explain why there is a pole located in 1 (an integrator pole) and one pole outside the unit circle – how do they relate to the dynamics of the real robot? □

We will use diagonal weight matrices for our LQ design. Suitable first guesses of weight matrices are

$$Q_1 = \begin{bmatrix} \frac{1}{m_1^2} & & & \\ & \frac{1}{m_2^2} & & \\ & & \frac{1}{m_3^2} & \\ & & & \frac{1}{m_4^2} \end{bmatrix} \quad Q_2 = \frac{1}{m_u^2}$$

where m_i and m_u are approximate magnitudes of the intended working ranges of the states and control signal. For example, it is reasonable to assume that we can only tolerate a small working range for the tilt angle, α (otherwise the MinSeg will fall). A reasonable working range, m_2, could then be 0.1°—1°. Remember to convert everything to radians. Try to visualize what would happen to the weights if you increased/decreased the working range.

Assignment 11 (Preparatory). Come up with initial choices of m_i and m_u. The control signal, u, cannot exceed 3.25 V (the typical working range should thus be much smaller) and the robot cannot recover from a tilt angle, α, greater than a few degrees ($\approx 0.1°$—$2°$). For m_1, i.e., the working range for $\dot{\alpha}$, the MinSeg should be able to go from the working range limit of α to $\alpha = 0$ in approximately 0.01—0.1 s. The wheel position θ is not as crucial to penalize, an acceptable working range could then be what? The same holds for the wheel velocity $\dot{\theta}$. **Do not forget to motivate your choice.** □

Assignment 12 (Preparatory). Find the LQ controller, `feedback_gain`, by utilizing the discretized system matrices (`Phi` and `Gamma`) from Assignment 10 and the initial weight matrices (`Q1` and `Q2`) from Assignment 11 in Matlab.

Simulate the closed-loop response to the initial condition $\alpha = 0.04$ rad (all other states zero) by using the function `initial_sim_cl.m` (provided to you in the zip file).

```
feedback_gain = dlqr(Phi, Gamma, Q1, Q2)
initial_sim_cl(Phi, Gamma, feedback_gain, Ts)
```

The plot shows the response of the four states as well as the control signal (in the fifth subplot). The tilt angle should recover within about 1 s, while the wheel angle could take much longer to recover. The control signal magnitude should not exceed 3.25 V. Adjust the design weights and repeat the above procedure until you have a controller that seems reasonable. □

Assignment 13. Check if the IMU needs to be recalibrated and do so if needed.

For the controller that performed well in Assignment 12; run the Simulink model and test the controller on the MinSeg. Finally, turn the Toggle Control Action on to activate the controller. To stop the controller, toggle the same switch back to Off. **The MinSeg is very sensitive, help it by first balancing it by hand in the upright position** and make sure that the control signal u looks reasonable. Once it seems to balance by itself, you are free to let it do so by toggling the **Toggle Control Action** switch.

Does it work? If yes, see whether you can improve the behavior by playing with the design weights.

Common problems if it is unable to balance:

- Check if the IMU needs to be recalibrated.
- Check that the wheels are not rubbing against the body of the MinSeg.
- The MinSeg is really sensitive so despite your LQ-controller being near perfect, it might need your help during an initial phase.

□

8. Integral Action and Reference Tracking (Optional)

Once the robot is able to balance on its own, we want to control its position to a specific reference for the wheel angle θ (the reference is a smoothed square wave in the Simulink diagram). However, we have no way of specifying this requirement in our original LQ problem; another state is thus added to our model. We add the new state as an integrator

$$\begin{aligned} \hat{x}_i = \int (r - \hat{x}_4)\, dt \quad &\xrightarrow{\text{ZOH}} \quad \hat{x}_{i,k} = \frac{h}{z-1}\,(r_k - \hat{x}_{4,k}) \\ &\implies \hat{x}_{i,k+1} = \hat{x}_{i,k} + h \cdot (r_k - \hat{x}_{4,k}) \end{aligned} \tag{1}$$

where $\hat{x}_4$ is the estimated wheel position from the second Kalman filter. Since it is desirable that the error always goes to zero (regardless of the disturbance), $\hat{x}_i$ is added as an integrator in order to integrate the error a way.

We will now extend our model to include $\hat{x}_i$ and design a new LQ feedback for our expanded state space model. Because this is a common thing to do, there exists a specific command in Matlab to do this, `lqi`.

The new integrator state is however not present in the real process. Therefore, the integration needs be performed in the controller itself. In the Simulink model an error signal is fed into the Control law block where the error is integrated (and also reset, in case of some failure).

Assignment 14. Use `integral_extension.m` to design an extended state feedback vector $K_e = [\, K \quad k_i \,]$ that can be used in the extended control law

$$u_k = -K\hat{x}_k - k_i\hat{x}_{i,k}.$$

Proceed in Matlab as follows:

```
mi = ...;
Qi = 1/mi^2;
[feedback_gain,integral_gain] = integral_extension(Phi,Gamma,Q1,Q2,Qi,Ts)
```

Run the Simulink model and stabilize the MinSeg. Toggle the Toggle Integral Action switch to On and make sure that the MinSeg is still stable. If the system seems stable, activate the reference generator by inputting a suitable amplitude (e.g., π). Does it work well? Does it suffer from wind up? If needed, go back and tune the Kalman filters or the state feedback further. □

Note that if the MinSeg would fall while the integral action is turned on, the wheel angle will have to be reset. This is done by turning the integral action off, turning the control action off, standing the MinSeg up, and flicking the Toggle Reset Reference whilst helping the MinSeg stabilize manually. Thereafter it is possible to turn the controller on again.

9. Summary and Evaluation

Assignment 15. Answer the following questions (motivate).

1. What do the R_1, R_2, and R_{12} matrices represent?

2. Name at least two things important to remember when choosing weighting matrices Q_1 and Q_2.

3. What does the bandwidth of the Kalman filter tell us? What would have happened if we chose a lower bandwidth to aim for?

4. Name some advantages of *optimal* Kalman filters and LQ controllers in comparison to classic observers and state feedback.

5. Why did we introduce integral action on the wheel position?

□

References

[1] *Angle estimation using gyros and accelerometers (lab PM)*, January, 2018. Division of Automatic Control, ISY, Linköping University, Sweden.

[2] Brian Howard and Linda Bushnell. Enhancing linear system theory curriculum with an inverted pendulum robot. In *Proc. American Control Conference*, 2015.