FRTN10 Multivariable Control

Laboratory Session 3

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

Department of Automatic Control
Lund University

1. Introduction

In this laboratory session we will develop Kalman filters and a linear-quadratic (LQ) controller for the MinSeg™ balancing robot, see Figure 1. The MinSeg is based on a Genuine Arduino Mega 2560 microcontroller and is equipped with a Lego NXT DC motor with axles, wheels and encoder, as well as a 3-axis combined gyroscope and accelerometer. A USB connector allows the Arduino to be programmed from a PC and also enables the reading of plot data and writing of parameters during runtime.

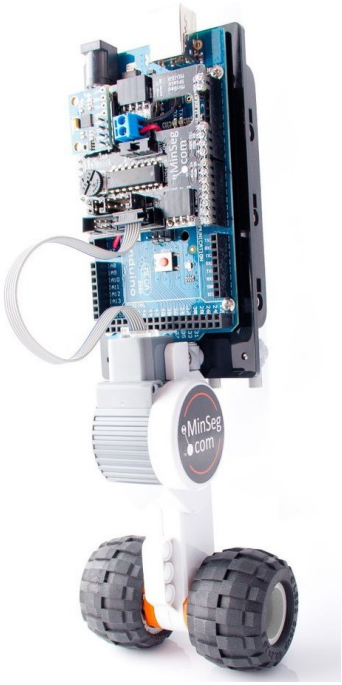

Figure 1 The MinSeg™ balancing robot.

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

Pre-lab assignments

Review lectures 9–11 on LQ control, Kalman filtering, and LQG control. Complete the Preparatory assignments 5, 7, and 9. Read this entire document carefully.

[1] Written by Anton Cervin, latest update October 5, 2018.

2. The Lab Interface

We will use Simulink with Simulink Coder (formerly Real-Time Workshop) for modeling and implementation of the filters and controllers. The main diagram is shown in Figure 2.

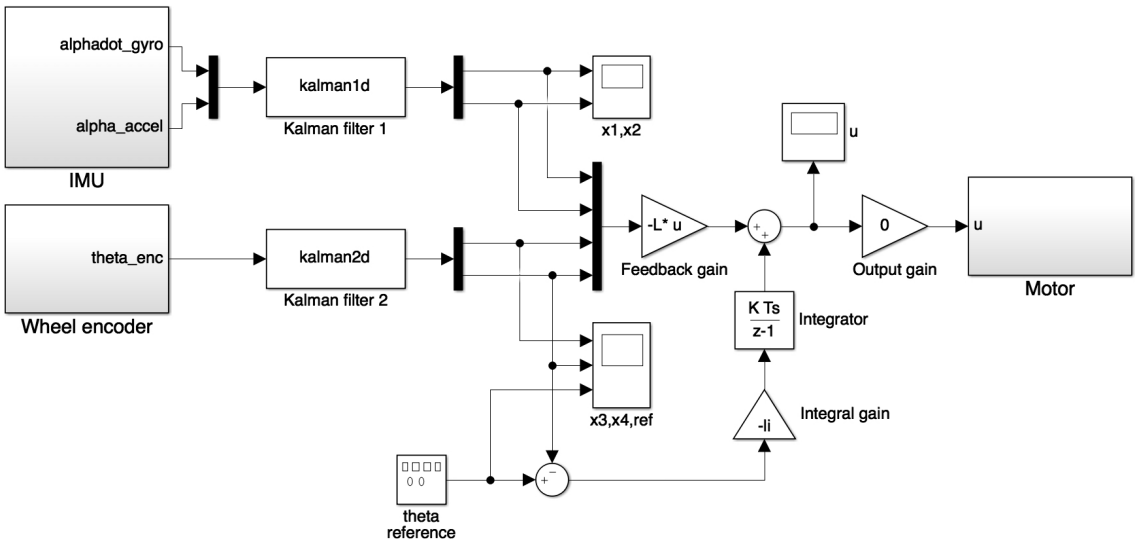

Figure 2 Simulink model `lab3.slx` for filter and controller implementation.

The Simulink model runs in discrete time with the sampling interval `Ts` (defined in the Matlab workspace). When balancing the robot we want to use a very short time interval, between 10 and 20 ms, in order not to introduce too much extra latency in the control loop. For such short intervals, however, Simulink might not have time to update the scopes properly. In the first assignments, it is therefore better to use a longer sampling interval, between 30 and 50 ms.

Assignment 1. Download `lab3_files.zip` from the course homepage and extract the contents to some suitable working directory. In a terminal window, type

```
VERSION=R2016a matlab
```

to start Matlab R2016a. Once Matlab has started, cd to the `lab3_files` directory and then type

```
setup_lab3
```

to setup the paths to the Matlab/Simulink support packages and the Simulink libraries for the Arduino and MinSeg hardware.

Open up and explore the Simulink model `lab3.slx`. When the model is opened, default (zero) values for the controller variables `kalman1d`, `kalman2d`, `L`, and `li` are automatically defined in the Matlab workspace.

Make sure that you understand how the IMU, Wheel encoder and Motor blocks relate to the real MinSeg robot. Also check the current value of `Ts`. □

3. Process Overview. Calibration of Measurement Signals

Mechanically, the MinSeg consists of two main parts: the body and the wheels. They are linked via the DC motor, a gearbox, and the wheel axles. Figure 3 shows the coordinate system we will

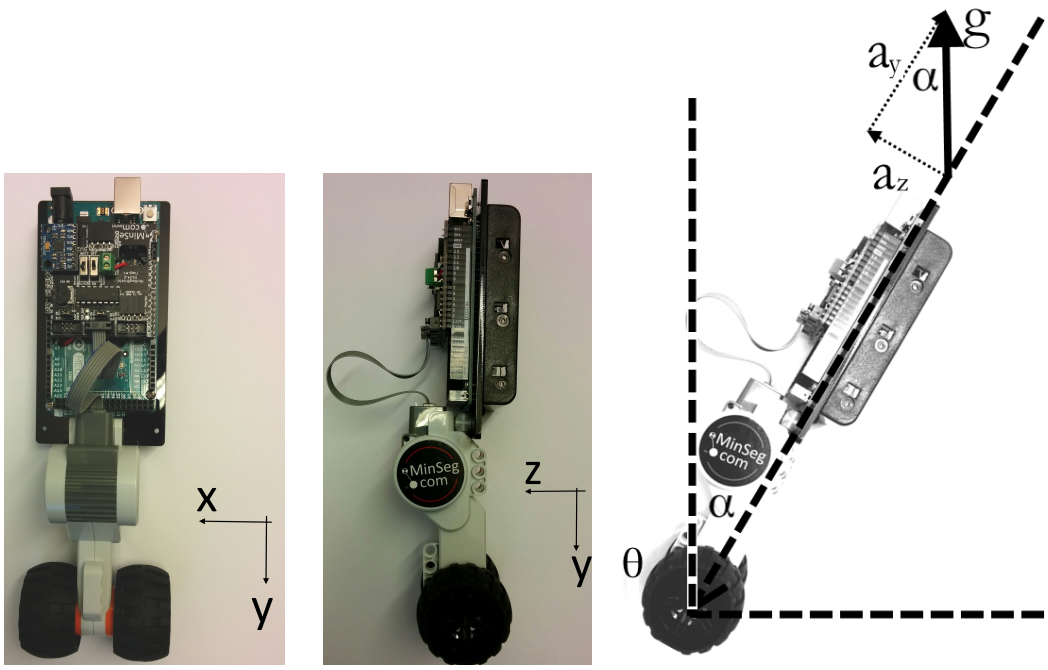

Figure 3 Definition of x, y, and z axes, tilt angle α, and wheel angle θ (adapted from [1]).

use. The tilt angle of the robot is denoted α and the wheel angle is denoted θ. In the lab we will estimate and control the four state variables $\dot{\alpha}$, α, $\dot{\theta}$, and θ. Modeling of the dynamics of the robot will be discussed later, in Section 6.

The MinSeg is equipped with a digital inertial measurement unit (IMU, the blue chip marked MPU6050) that contains a triple-axis gyro and accelerometer. The gyro measures movement (angular velocity) around the x, y, and z axes. In our application, the x gyro signal is proportional to $\dot{\alpha}$, i.e., it measures the tilt rate of the robot. The signal is however noisy and has a slowly drifting offset.

The accelerometer measures the acceleration of the IMU relative to free fall. 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 3 and calculate the tilt angle according to

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

The accelerometer signals are noisy and have slowly varying offsets. They will 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).

Assignment 2. Connect the MinSeg to the computer using the USB cable. Check that the COM port is properly defined in the Simulink model under Simulation / Model Configuration Parameters / Hardware Implementation / Host-board connection. (Switching from Manually to Automatically and back to Manually again normally sets it right.)

Click "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.) When the model is running, open up the IMU subsystem 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. □

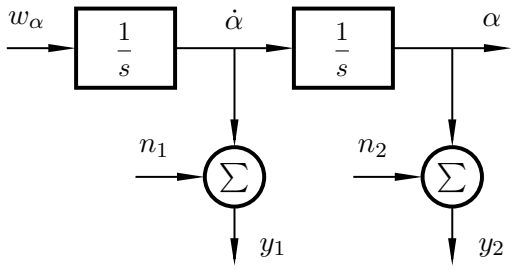

Figure 4 Model of the IMU for design of the first Kalman filter.

Assignment 3. Lay the robot flat on its back (battery case towards the table) and calibrate the x gyro and y accelerometer readings to zero (approximately, on average) by entering suitable values for the offsets xvel_bias and yaccel_bias. Then stand the robot up as close to its balancing point as possible and calibrate the z accelerometer reading to zero by similarly adjusting zaccel_bias. Check that the calculated α value (alpha_accel) correctly describes the robot tilt angle in stationarity. □

The calibration procedure described above should be repeated each time a new run is started, since the raw values of the gyro and the accelerometer tend to drift.

Assignment 4. Keep the robot completely still for $1000 \cdot$ `Ts` seconds and then hit "Stop". The 1000 most recent datapoints of alphadot_gyro and alpha_accel are automatically stored in the workspace. Plot the signals, remove any linear trends, calculate their variance, and plot their spectra, using, e.g.:

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

Can the signals reasonably well be modeled as white noise? □

4. Design of Kalman Filter for $\dot{\alpha}$ and α

We start by designing a Kalman filter that uses the angular velocity measurement y_1 from the gyro and the calculated angle measurement y_2 from the accelerometer to estimate the IMU angular velocity $\dot{\alpha}$ and angle α. Ignoring the rest of the robot, the system can be modeled as a double integrator from the external angular acceleration w_α (representing the external forces on the chip) to the tilt angle α, see Figure 4.

Assignment 5 (Preparatory). Convert the model in Figure 4 to state-space form using the state vector $\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} \dot{\alpha} \\ \alpha \end{smallmatrix}\right)$. 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.

Setting $R_2 = I$, write down the algebraic Riccati equation and the resulting set of quadratic equations involving the elements of the error covariance matrix $P = \left(\begin{smallmatrix} p_1 & p_2 \\ p_2 & p_3 \end{smallmatrix}\right)$. Would it be easy to solve these equations by hand?

Using `lqe` in Matlab, calculate the Kalman filter gain K and the resulting observer poles for some different values of R_1 (very large and very small). How is the relative size of R_1 compared to R_2 influencing the speed of the observer? □

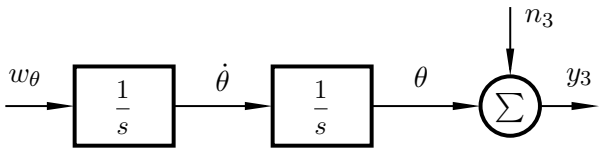

Figure 5 Model of the wheels for design of the second Kalman filter.

Assignment 6. Design the Kalman filter for $\dot{\alpha}$ and α in Matlab, using the measured values of R_2 from Assignment 4 and some arbitrary value for R_1 as a starting point. Work in a Matlab script so that you can easily repeat the whole procedure. Using `ss`, formulate the Kalman filter as a state-space system `kalman1` according to

$$\begin{aligned}\frac{d\hat{x}(t)}{dt} &= (A - KC)\hat{x}(t) + Ky(t)\\ \hat{x}(t) &= I\,\hat{x}(t)\end{aligned}$$

The system should have two inputs and two outputs to match the Simulink model. Finally, convert the filter into a discrete-time system `kalman1d` using `c2d` and first-order hold sampling[1] as follows:

```
kalman1d = c2d(kalman1,Ts,'foh');
```

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?

Hit "Run" and try the Kalman filter on the real process. After calibrating the measurements, tilt the robot by hand and observe how fast the estimates $\hat{x}_1$ and $\hat{x}_2$ are following the movements. Hit "Stop", repeat the whole procedure with different design matrices and observe the difference in tracking speed. For balancing, the filter bandwidth from y_1 to $\hat{x}_1$ should be at least 50 rad/s and from y_2 to $\hat{x}_2$ about 1 rad/s. □

5. Design of Kalman Filter for $\dot{\theta}$ and θ

Next, we turn to designing a second Kalman filter for estimating the wheel angular speed $\dot{\theta}$ and position θ from the encoder measurement y_3. A model of the subsystem is shown in Figure 5. The measurement noise n_3 represents the quantization error of the wheel encoder, which has a resolution of 0.5 degrees.

Assignment 7 (Preparatory). Convert the model in Figure 5 to state-space form using the state vector $\left(\begin{smallmatrix} x_3 \\ x_4 \end{smallmatrix}\right) = \left(\begin{smallmatrix} \dot{\theta} \\ \theta \end{smallmatrix}\right)$. Assuming the relative noise intensities

$$R_1 = \begin{pmatrix} \omega^4 & 0 \\ 0 & 0 \end{pmatrix}, \quad R_2 = 1,$$

show that the algebraic Riccati equation for the Kalman filter has the solution

$$P = \begin{pmatrix} \sqrt{2}\,\omega^3 & \omega^2 \\ \omega^2 & \sqrt{2}\,\omega \end{pmatrix}$$

and that the resulting observer poles are given by the characteristic equation

$$s^2 + \sqrt{2}\,\omega s + \omega^2 = 0.$$

(We have hence shown that, for this problem, placing the two observer poles in the standard pattern with $\pm 45°$ angle from the negative real axis is optimal.) □

[1] You can learn more about discretization and implementation methods in FRTN01 Real-Time Systems.

Assignment 8. Design the Kalman filter for $\dot{\theta}$ and θ in Matlab. Aim for a filter bandwidth of at least 50 rad/s. Formulate the Kalman filter as a state-space system `kalman2` using `ss` (see Assignment 6). The system should in this case have one input and two outputs to match the Simulink model. Then convert it into discrete time using

```
kalman2d = c2d(kalman2,Ts,'foh');
```

Finally, hit "Run" and try the Kalman filter on the real process. Turn the robot wheels by hand and verify that the estimates $\hat{x}_3$ and $\hat{x}_4$ seem to behave as expected. □

6. Design of LQ State Feedback

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 9 (Preparatory). Calculate the poles of the linear robot model using Matlab. The fastest pole is related to the motor dynamics. Explain why there is a pole located in the origin—how does it relate to the real robot? Are there any fundamental limitations imposed by the dynamics? □

Assignment 10. Define the system matrices as well as some initial values of the design weights `Q1` and `Q2` in Matlab and then calculate an LQ controller for the robot using `lqr`:

```
L = lqr(A,B,Q1,Q2)
```

Simulate the closed-loop response to the initial condition $\alpha = 0.04$ rad (all other states zero):

```
syscl = ss(A-B*L,[],[eye(4);-L],0);
x0 = [0 0.04 0 0];
initial(syscl,x0);
```

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. At the same time, 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.

Before the real balancing test, set `Ts` to a smaller value and redo the discretization of the `kalman1` and `kalman2` systems.

Hit "Run" and test the controller on the robot. Calibrate the sensor readings, balance the robot by hand in the upright position, and check that the control signal u looks reasonable. Finally set the Output gain to 1 to activate the controller. DOES IT WORK?

To stop the controller, set the Output gain back to 0 and hit "Stop". □

7. Integral Action and Reference Tracking

In the final part of the lab we will introduce tracking of the wheel position reference signal r using explicit integration. The integrator state is given by

$$\dot{x}_i = r - \hat{x}_4$$

where $\hat{x}_4$ is the estimated wheel position from the second Kalman filter. As you can see, this calculation has already been implemented in the Simulink model.

Assignment 11. Use `lqi` to design an extended state feedback vector $L_e = \begin{pmatrix} L & l_i \end{pmatrix}$ that can then be used in an extended control law

$$u = -L\hat{x} - l_i x_i$$

Proceed in Matlab as follows:

```
Qi = ...                    % Integral state penalty
Q1e = blkdiag(Q1,Qi)        % Extended Q1 matrix
sys = ss(A,B,[0 0 0 1],0)   % Define system with theta as the only output
Le = lqi(sys,Q1e,Q2)        % Calculate extended feedback gain vector
L = Le(1:4)                 % Extract L
li = Le(5)                  % Extract li
```

Hit "Run" and test the controller. If the system seems stable, activate the theta reference by entering a suitable Amplitude value like π. If needed, go back and tune the Kalman filters or the state feedback further. □

8. Summary and Evaluation

Assignment 12. Answer the following questions on a piece of paper and hand in:

1. Write down the most important lessons learned by designing "optimal" filters and controllers for the MinSeg robot.
2. This was the first time this lab was given. What could be improved for future editions?

□

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.