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.

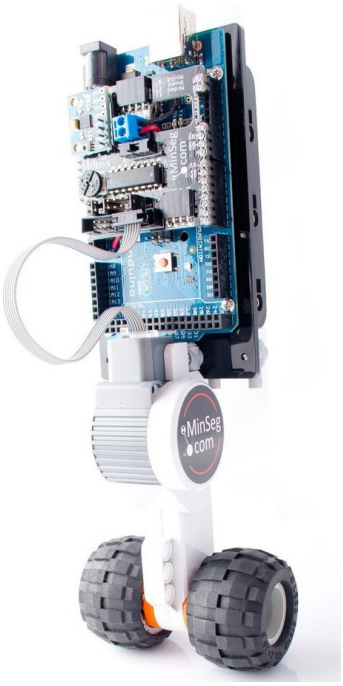

Figure 1 The MinSeg™ balancing robot.

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

Read this document and complete all assignments marked as *(Preparatory)*. Helpful lectures to review are lectures 9–11 on LQ control, Kalman filtering, and LQG control. Parts of lecture 3 concerning stochastic processes and their spectrum are also useful.

[1] Written by Anton Cervin, latest update September 30, 2019.

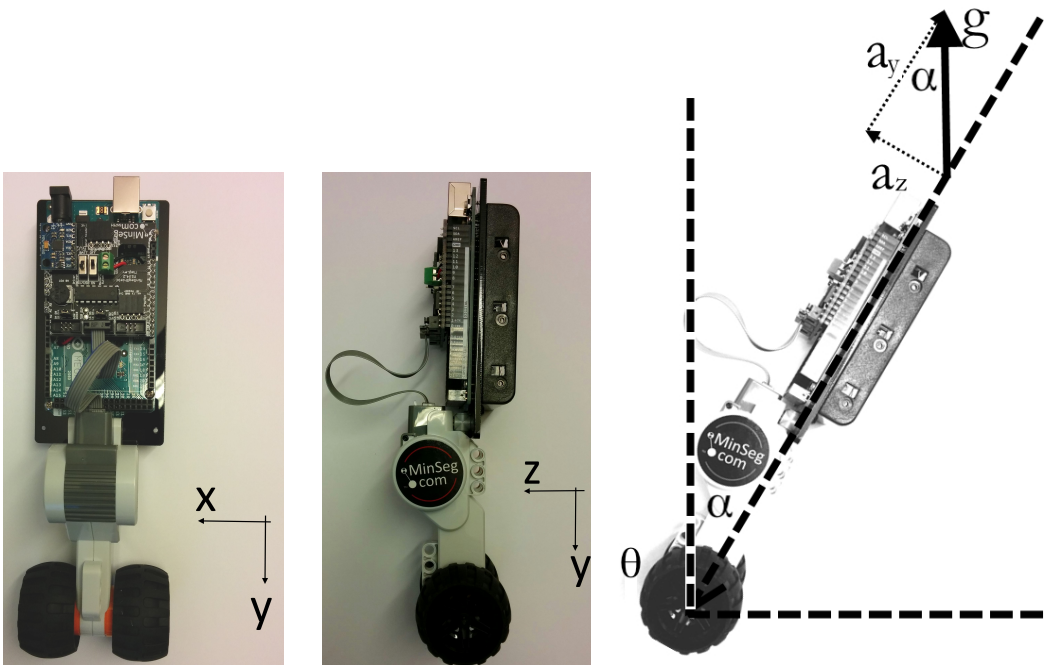

Figure 2 Definition of x, y, and z axes, tilt angle α and wheel angle θ (adapted from [1]).

2. The Process

The drivetrain of the MinSeg 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 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 needs to be found. Without going into details of the dynamics of the system, we say that the state consists of four state variables: The angle of rotation of the wheels, θ, and the tilt angle, α, as well as their time derivatives. Since we only measure two of these states directly, θ and $\dot{\alpha}$, and both of those measurements are noisy, the 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 (formerly Real-Time Workshop) for modeling and implementation of the filters and controllers. The main diagram is shown in Figure 3.

The model is configured by five variables in the Matlab workspace; these variables are `IMU_Kalman_D`, `Wheel_Kalman_D`, `Feedback_Gain`, `Integral_Gain`, and `Ts`. When the simulink

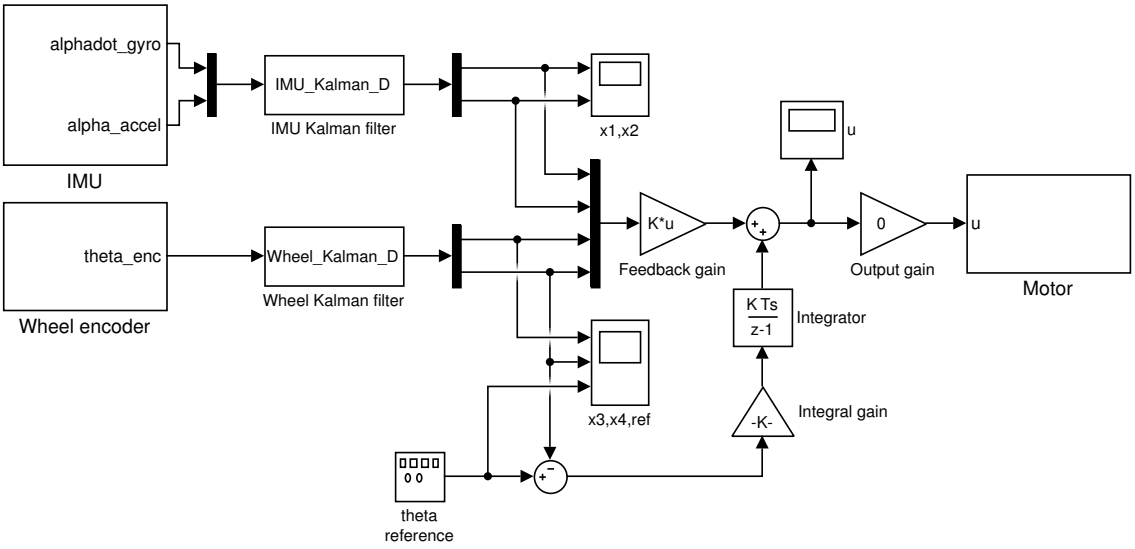

Figure 3 Simulink model `lab3.slx` for filter and controller implementation.

model is run, it will read these variables, compile the resulting controller and upload it to the MinSeg. To design our controller we simply assigning different kalman filters and feedback gains to the workspace variables. When the model is first opened, default (zero) values 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 25 ms. This value is a trade-off between control performance and plotting speed.

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, go into 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. **Please note:** It's important that you run this command directly after starting Matlab, and then never again.

Open up and explore the Simulink model `lab3.slx`. Make sure that you understand how the IMU, Wheel encoder and Motor blocks relate to the real MinSeg robot. Furthermore, make sure the current value of `Ts` is 25 ms. □

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.

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. □

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 balance the robot in the upright position as well as you can and calibrate the z accelerometer reading to zero by similarly adjusting zaccel_bias.

Make sure you zoom in enough. The biases need to be set within an accuracy of at least 25-50. □

The raw values of the gyro and the accelerometer can drift slightly during operation, so the calibration procedure might be needed to be repeated later during the lab session.

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

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

The accelerometer signals are noisy and they 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 4. Look at the alpha_accel scope in the IMU block of the Simulink model. Does the measurement correctly describe the tilt angle when the MinSeg is stationary? Move the MinSeg forward and backward without tilting it. Does the measurement of the angle seem correct? □

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 5. Keep the robot completely still for $1000 \cdot$ `Ts` seconds and then hit "Stop". The 1000 most recent measurements are automatically stored in the workspace in the variables of alphadot_gyro and alpha_accel. For the first measurement (alphadot_gyro), plot the signals, remove any linear trends, calculate their variance and save it in variable, and plot their spectra, using:

```
plot(alphadot_gyro)                    % plot
y1 = detrend(alphadot_gyro,'linear'); % remove linear trend
plot(y1)                               % plot again
pwelch(y1)                             % plot periodogram (estimate of spectrum)
y1var = var(y1);                       % calculate stationary variance
```

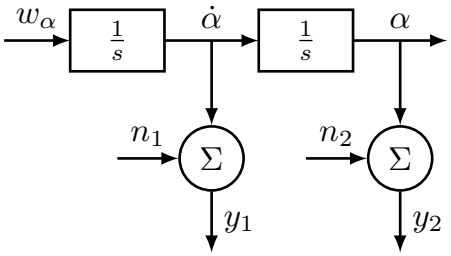

Figure 4 Model of the IMU for design of the first Kalman filter.

(Work in a Matlab script for this and all other assignments so the 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. Make sure to record the values of `y1var` and `y2var` for later use. □

5. Tilt Angle Estimation

We will use a Kalman filter to reduce the effect of the noise on our measurements y_1 and y_2 of $\dot{\alpha}$ and α. Kalman filters need a model of the dynamics, and 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 won't be as good as it could be, but it will be adequate for this application.

Assignment 6 (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? □

Assignment 7. Design the Kalman filter for $\dot{\alpha}$ and α in Matlab, using the measured values of R_2 from Assignment 5 and some arbitrary value for R_1 as a starting point. Using `ss`, formulate the Kalman filter as a state-space system `imu_kalman` 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 in order to match the Simulink model.

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 at least 50 rad/s and from y_2 to $\hat{x}_2$ about 1 rad/s.

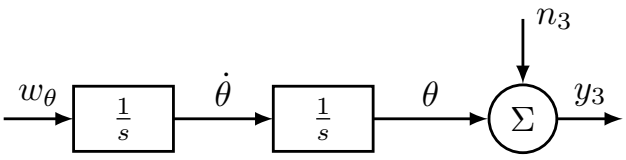

Figure 5 Model of the wheels for design of the second Kalman filter.

Finally, convert the filter into a discrete-time system `IMU_Kalman_D` using `c2d` and first-order hold sampling[1] as follows:

```
IMU_Kalman_D = c2d(imu_kalman, Ts, 'foh');
```

"Run" the Simulink model and try the Kalman filter 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. Hit "Stop", repeat the whole procedure with different design matrices and observe the difference in tracking speed. □

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 speed $\dot{\theta}$ and position θ. For this we will utilize the rotational encoder of the motor which will give us our 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 response of the motor on changes in the applied voltage and the inertia of the robot. A model of the subsystem is shown in Figure 5.

Unlike earlier, we now only have one measurement, the output from the encoder. This signal is not very noisy but is quantized with a resolution of 0.5 degrees, resulting in uncertainties. The measurement noise of the model, n_3, represents this quantization error of the wheel encoder.

Assignment 8 (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

$$R_1 = \omega^4, \quad R_2 = 1,$$

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.) □

Assignment 9. 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 `wheel_kalman` 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 and save it to `Wheel_Kalman_D` using

```
Wheel_Kalman_D = c2d(wheel_kalman, 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. □

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

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). 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 dynamics of the real robot? Are there any fundamental limitations imposed by the dynamics? □

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 the rough magnitudes of the intended working ranges of the states and control signal. For example, the maximum speed of the wheels, $\dot{\theta}$, is around 2π rad/sec, a reasonable working range, m_3, must be smaller than that or the controller might be to aggressive, resulting in saturation problems. On the other hand, too small of a working range would result in a controller unwilling to make the fast movements necessary to balance.

Assignment 11 (Preparatory). Come up with initial choices of m_i and m_u. The control signal, u, can't exceed 3.25 V and the robot can't recover from a tilt angle, α, greater than a few degrees (≈ 0.05 rad). For the m_1, i.e. the working range for $\dot{\alpha}$, use your knowledge of the poles of the system and base it on your working range of α.

Assignment 12. Define the system matrices with your initial values of the design weights `Q1` and `Q2` in Matlab and then calculate an LQ controller for the robot using `lqr`:

```
Feedback_Gain = 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*Feedback_Gain,[],[eye(4);-Feedback_Gain],0);
x0 = [0 0.04 0 0];
initial(syscl,x0,1);
```

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. □

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

For the controller that performed well in the simulations; run the Simulink model and test the controller on the MinSeg. Balance the robot by hand in the upright position and make sure that the control signal u looks reasonable. Finally, set the Output gain to 1 to activate the controller. To stop the controller, set the Output gain back to 0 and hit "Stop".

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 wheel are not rubbing against the body of the MinSeg.

□

8. Integral Action and Reference Tracking

Now that we have managed to balance the robot, we want to control its position on the table by specifying a reference for the wheel angle θ. (The reference is a square wave in the Simulink diagram.) However, we have no way of specifying this requirement in our original LQ problem, so to do this we need to add another state to our model. We add the new state as an integrator

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

where $\hat{x}_4$ is the estimated wheel position from the second Kalman filter. x_i is added as an integrator since we always want the control error going to zero, regardless of disturbances.

We can now extend our model to include x_i and design a new LQ feedback for our expanded state space model. Matlab has a specific command, `lqi`, that does this for all the outputs of a state space model, since this is a common thing to do. However, the new integrator state is not present in the real process. Therefore, the integration needs be performed in the controller itself. In the Simulink model a block that integrates this error has already been included.

Assignment 14. Use `lqi` to design an extended state feedback vector $L_e = \begin{pmatrix} L & l_i \end{pmatrix}$ that can be used in the 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
Feedback_Gain = Le(1:4)     % Extract L
Integral_Gain = 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 for the square wave like 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. □

9. Summary and Evaluation

Assignment 15. Answer the following questions and hand in this sheet of paper (or use a separate piece of paper):

1. Write down the most important lessons learned by designing "optimal" filters and controllers for the MinSeg robot.

2. This was the second 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.