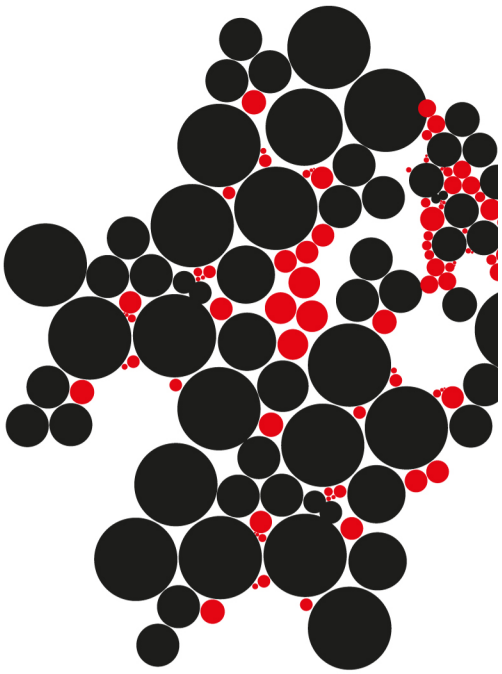

LAB MANUAL

Original: T.G. Broenink, W.Horlings
2019 revision: M. Schouten

EEMCS
Module 6: Systems and Control

UNIVERSITY OF TWENTE.

JANUARY 11, 2020

Contents

Chapter 1

Introduction

This manual covers the project and pre-projects of module 6: Systems and Control. There are four pre-projects to prepare you for the project. The project consists of two parts. First, the motor measurement part. Second, the segway control part.

At the end of this project you are able to model and control a dynamic system, which is a small segway for this project. Essential skills are model design, parameter estimation, and design and tune of controllers.

The grade of this project will depend on two reports, one for the motor and one for the segway project. Furthermore, one bonus point can be earned with the performance of the segway during the final challenges.

1.1 Organization

Both the pre-project as the project will be done in groups of 2. You will have to write a report on both the motor measurement part and segway control part.

1.1.1 Planning

The pre-projects will be on the Fridays at the beginning of the quartile. Each will take three quarters of a day.

The motor measurement part is done in a total of 2.5 days, which can be split over multiple days. For the segway project 9 days are available. Halfway the project you will have to present your Segway model to the TA's. At the end of the project there will be a demonstration where you can show off the performance of the controller of your segway. The segway project uses the motor model you made during the first part of the project.

Deadlines:

- Tuesday 7 January 2019 (13:30 GMT+1): Submit motor report
- Monday 20 January 2019 (all day): Segway model presentation
- Friday 24 January 2020 (afternoon): Segway demonstration
- Sunday 26 January 2020 (23:59 GMT+1): Submit Segway report

1.1.2 Groups

The pre-projects are done in groups of 2. These groups should remain the same for all the pre-projects. You have to complete the pre-projects to be able to participate in the project.
The project is split into two parts and is also done in groups of 2. The project groups may be different from the pre-project groups. Your group will be assigned to a segway for the project. As multiple groups share one segway, there will be time slots for measurements. Please make sure both your pre-project as your project group is registered on Canvas.

1.1.3 Journal

Each student has to keep a logbook of their work. You do not have to hand it in and it will not be graded. However, it can be very useful, for example to be able to remember exactly what you did that one time your segway was super stable. Or to show a SA what you changed before your segway stopped working. This is especially useful during the project where you will be working 9 days with models and controllers. A logbook with what you did during each measurement and in each of your simulations makes writing a report easier.

1.1.4 Sign-off queue

To sign-off the pre-projects you will have to add your group number to the sign-off queue on `horusapp.nl`. This is necessary because we don't have the capacity to sign-off everybody at the same time. This queuing list will allow you to continue with the next exercise at your table instead of having to stand in line.

1.2 Grading

The pre-projects will have to be completed before you can participate in the project. You will have to show the specific deliverables of the pre-project to a SA, who will sign off on your work.

For your final grade, the motor measurement report will count for 30% and the project report for 70%.

1.2.1 Motor Measurement

The grade for the motor measurement will be based on the report and the observations from the SA. The focus of this report should be on the following points.

- How is your bond graph model structured? Why did you include or left out specific elements?
- How does your model compare with the actual behaviour? How did you determine the parameters?
- What kind of controller did you choose to implement? How did you tune the controller?

1.2.2 Segway

The grade for the Segway control will be based on the report and the observations from the SA. The focus of the report should be on the following points.

- How was the Segway model created? Why did you include or exclude specific elements?
- How were the Segway parameters determined, and more importantly, how were these validated?
- What design choices did you make when designing the segway controller, and why?
- How were the Segway controllers calculated and tuned?
- How does the final system perform, how did you validate this?

1.2.3 Responsibility and fraud

All group members share responsibility for the work handed in. When you distribute work among group members, check each other's work, discuss the work among all group members, do you understand it all?

You must submit original work.

Plagiarism, i.e. copying of someone else's work without proper reference, is a serious offence which in all cases will be reported to the Examination Board. Refer to the Student Charter (2017).

In cases where a non-trivial amount of work is copied with proper reference, indicate which parts are copied and which parts are your own original work. The copied work will not be considered for grading. If you use documents, measurements of models provided during this course, you do not have to mention that.

What constitutes fraud?

When it comes down to handing in assignments, every year there are students who do not understand the borderline between, on the one hand, cooperating and discussing solutions between groups (which is allowed), and on the other, copying or sharing solutions (which is forbidden and counted as fraudulent behaviour). Here are some scenarios which may help in making this distinction.

Scenario. Peter and Lisa are quite comfortable with programming and have pretty much finished the assignment. Mark andWouter, on the other hand, are struggling and ask Lisa how she has solved it. Lisa, a friendly girl, shows her solution and takes them through it line by line. Mark and Wouter think they now understand and go off to create their own solution, based on what they saw. Is this allowed or not?

Verdict. No problem here, everything is in the green. It is perfectly fine and allowed for Lisa to explain her solution, even very thoroughly. The important point is that in implementing it themselves and testing their own solution,Mark andWouter are still forced to think about what is happening and will gain the required understanding, though probably they will not get as much out of it as Lisa (explaining stuff to others is about the best possible way to learn it better yourself!).

Scenario. The start is as in the previous case. However, while Mark and Wouter implement their own solution, inspired by that of Lisa, some error crops up which they do not understand. Lisa has left by now; after they mail her, still trying to be helpful she sends them her solution for them to inspect. They inspect it so closely that in the end their solution is indistinguishable from Peter and Lisa's, except for the choice of some variable names and the comments they added themselves. Is this allowed or not?

Verdict. This is now a case of fraud. Three are at fault: Lisa for enabling fraud by sending her files (even if it was meant as a friendly gesture) andMark andWouter for copying the code. Peter was not involved, developed his own solution (together with Lisa) and is innocent.

Scenario. Alexandra and Nahuel are not finished, and the deadline is very close. The same holds for Simon and Jaco. On the Friday night train home, Jaco and Nahuel meet and during the 2-hour train ride work it out together. They type in the same solution and hand it in on behalf of their groups. Is this allowed or not?

Verdict. This is also a case of fraud. Actually there are two problems here. The first is that both Nahuel and Jaco handed in code on behalf of their groups that had been developed by them alone, without their partners. This is unwise and against the spirit of the assignment (Alexandra and Simon also need to master this stuff!) but essentially undetectable and not fraudulent. The second problem is that the solution was developed, and shared, in collaboration between two groups; this is definitely forbidden. All four students are culpable; Alexandra and Simon cannot hide behind the fact that they did not partake in the collaboration, as they were apparently happy enough to have their name on the solutions and pretend they worked on it, too.

Note that we are not on a witch-hunt here: let us stress again that cooperating and discussing assignments is OK, even encouraged; it is at

the point where you start copying or duplicating pieces of code that you cross the border.

Chapter 2

Pre-project 1: Modeling

2.1 Introduction

The goal of this pre-project is to familiarize you with simulating and analysing models in 20-sim. This will be done by a series of exercises based on modelling a system that will be extended a few times. In this manual you will find a series of questions/commands. Make sure you log all your answers and steps in your logbook as explained in subsection 1.1.3. When asking questions, a SA can ask you to show your logbook.

To complete this pre-project you need to show a SA your work, so that they can sign off. To sign-off you have to add your group number to the queuing list on the whiteboard, see section subsection 1.1.4. What you exactly need to show is defined in section 2.14.

2.2 20-sim

20-sim is the modeling and simulation software used in this module. For instructions on how to install it see Canvas. Furthermore you should read the 20-sim guide, which is also available on Canvas. This guide contains a getting started section. You can always refer back to this guide when looking for features.

- *Install and familiarize yourself with 20-sim using the 20-sim guide*

2.3 The system

The system that you will model is a water pump. This pump is used to fill a reservoir. The pump is connected with an electromotor via a belt. This system looks like Figure 2.1. We are interested in the water in the tank, based on the input provided by the motor.

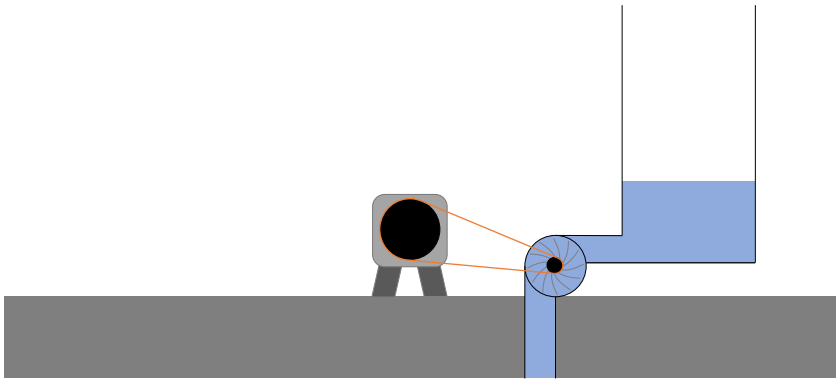

Figure 2.1: The system to model, a centrifugal pump driven by an electromotor via belt.

2.4 First model

You start the model of the system with a few simple models of the components in the system. Later in this assignment you will replace these by more complex/complete models. When looking at this system you can identify a few different parts:

- The motor
- The belt transmissions
- The pump
- The reservoir

When making a model of this system, keep these parts separate. For the first model we will keep the model as simple as possible. Assume that all component have a 100% efficiency and that all rotating components have a negligible mass and inertia.

- *What are the ideal bond-graph elements that can be used to model these 4 components?*

To model the system, you need a few system parameters:

- The power supply outputs 24 V.
- The motor has a torque constant (K_t) of $0.6\,\mathrm{N\,m\,A^{-1}}$.
- The transmission ratio of the belt is 4:1.
- The pump supplies $150\,\mathrm{Pa\,s\,rad^{-1}}$, which is pressure per rotational velocity.

- The reservoir has a surface area of $0.7\,\mathrm{m}^2$.

• *Make a bond graph model of this system using these elements and parameters. (Hint: The parameter of the reservoir is not in* m^2*, is probably smaller as you think and can be calculated from the equation for the pressure at the bottom of the tank when it contains a certain volume of water.)*

• *Run a simulation where you plot the electric current and the water level. (Note, the volume in the tank is not equal to the height of the water). Looking at the simulation results, is this a realistic model?*

2.5 Model organization

The current model is ideal. The electric current that is zero, and the instantaneous filling of the water tank are not realistic. Therefore, we will create a better model for each component. However, we do not want to overwrite the current ideal components. To do so, each components model will be in a separate submodel. The relevant components for which you want to create a submodel each, as mentioned before, are:

- The motor
- The belt transmissions
- The pump
- The reservoir

To split the current model into submodels, select the components in the model that you want to group; right-click on the selected components; and click *implode*.

- *Look up the implode-feature in the 20-sim guide*
- *Separate your model into the specified submodels.*

2.6 Water Reservoir

A reason why the system fills instant, is that the water is not subject to resistance. So a good first step to improve the model is to add a resistance to the inlet of the reservoir. As we made a submodel for the reservoir component we can add a new, more realistic, implementation of this component. 20-sim has a special feature for this. Right-click a submodel and select *Edit Implementations>Add New* and give your new implementation a name. With this menu you can to switch between your

different implementations. For more information on this see the 20-sim guide.

- *Create a new implementation for the reservoir.*

In this lab you are going to switch between different implementations. So make sure you use them. Different safe files for each model will not allow you to combine implementations at the end of this lab.

We found that the inlet-pipe of the reservoir has a resistance of $1200\,\mathrm{Pa\,s\,m^{-3}}$.

- *Add a resistance to the inlet of the reservoir and investigate how the electric current and water level changed in simulation.*

2.7 Transfer Function

Now the system does not respond instantly anymore. To show this, we can either do a plot of the voltage, or we can determine a transfer function of the system using 20-sim. This can be done with the model linearization tool, found under *Tools>Frequency Domain Toolbox>Model Linearization*. For instructions on how to use this tool please refer to the 20-sim guide. When plotting the step response of the system, make sure you simulate for long enough.

- *Generate the transfer function from voltage to water level using the Model linearization tool.Is this transfer function what you expected?*

2.8 Electric Current and Reservoir plot

There are different ways to view your simulation data. As we are going to change almost everything in the model we want something to show. By default 20-sim puts time on the x-axis. However, you can change that in the plot settings. More information about different plot options can be found in the 20-sim guide.

- *Create a new plot and plot the curves for the pressure and the water height in the reservoir against the electric source current.*

2.9 Belt Transmission

Now that we added the resistance, the electric current is unrealistically high and the settling time is very short. The belt transmission consists out of 2 wheels and a belt, which currently is probably modelled by a single transformer. We expect that this implementation is too simple. Therefore, we are going to take a look at the transmission. To improve

the model of this transmission, we have some information about these components:

- The first wheel, attached to the motor, has a 10 cm radius and a moment of inertia of $4.2 \times 10^{-3}\,\mathrm{kg\,m^2\,rad^{-1}}$.
- The second wheel, attached to the pump, has a 2.5 cm radius and a moment of inertia of $1.69 \times 10^{-5}\,\mathrm{kg\,m^2\,rad^{-1}}$.
- The wheels are connected with a elastic belt, which stretches by 3.5 µm for every Newton of force applied.

• *Create an alternative implementation of the transmission using this information.*

• *How does the step response of the system look now? And the transfer function?*
If you cannot find a difference in the step response, zoom in on the initial part of the response.
• *If there is an oscillation in the system, how is that represented in the plot you made in the last section?*

2.10 Centrifugal Pump

Now that the transmission has been improved, we will add detail to the pump. The pump, which was modeled as an ideal centrifugal pump, is less ideal than expected. The impeller of the pump has mass, and thus inertia. Furthermore, there is some friction in the pump axis and the pump leaks. The leak is in the outlet of the pump, which connected to the water reservoir. If we quantify these parameters we get the following information:

- The pump impeller has a moment of inertia of $1.6 \times 10^{-5}\,\mathrm{kg\,m^2\,rad^{-1}}$.
- The friction of the pump axis is approximately $5 \times 10^{-6}\,\mathrm{N\,m\,s\,rad^{-1}}$.
- The leak has a resistance of $35 \times 10^{6}\,\mathrm{Pa\,s/m^3}$.

When modeling the leak, take into account that this is an alternative path for the water to flow through, instead of into the reservoir.
• *Model an alternative model of the pump using the given information.*

• *Run the simulation again with the changed pump, what are the differences that you observe?*

2.11 Electro motor

Switching on the systems still results in an immense electric current. The current motor has no internal resistance or inductance. Which is not

very likely. After some improved measurements we found the following parameters:

- The motor coil has a resistance of $2\,\Omega$.
- The motor coil has an inductance of 160 mH.

• *Improve your motor model. What effect does this have on the transfer function of your system? And what effect does it have on the step response?*

• *The steady state value of your pump system is now different, or at least it should be. What do you think caused this?*

2.12 Model reduction

You now have three more detailed models, but are they all necessary? Maybe some of the implementations have not a significant influence on the final model. To determine this, we will observe the change for each sub-model once more. Make sure that if one sub-model is on the original implementation all the others are on their improved (detailed) implementation.

• *Create a table with the effects of the simplifying the sub-models, also think about why the effects happen.*

This table should look like:

Simplified model	Effect	Why?
Transmission	*observed effects*	Probable reason
Motor	...	...
Pump	...	...

• *If you want to use the model of your pump system to create a controller for the water level, which model would you use and why? What is the transfer function?*

2.13 Extra fun

If you want to do some extra fun exercise, which is not required for this assignment, but might be fun to do. Try to change your tank model in such a way that the tank is only 2 meters high. Thus it will overflow if the water goes above this level.

2.14 What to show

You do not have to show all your answers/model to the SA's to complete this pre-project. You should show the following to your SA:

- A 20-sim file in which you can switch your model implementations, make sure it is nicely formatted. Formatting guidelines can be found in the 20-sim guide.
- The transfer function of the following situations, from input voltage to water level:
  - The simple model.
  - The model with detailed transmission
  - The model with detailed transmission, motor and pump.
  - The model with the detailed motor and pump, but not the transmission.
- The table described in section 2.12.
- The final model you would use to design a water level controller.

Remember that your written journal is your proof of work. If it is not written down, you did not do it.

Chapter 3

Pre-project 2: Control

1System X

3.1 Introduction

The goal of this pre-project is to familiarize you with controlling a given system using 20-sim. This will be done by a series of exercises based on a given system. In this manual you will find a series of questions/commands. Make sure you log all your answers and steps in your logbook as explained in subsection 1.1.3. When asking questions, a SA can ask you to show your logbook.

To complete this pre-project you need to show a SA your work, so that they can sign off. To sign-off you have to add your group number to the queuing list on the whiteboard, see section subsection 1.1.4. What you exactly need to show is defined in section 3.7.

3.2 Given System

During this pre-project you are going to develop a controller for a system, which we will call System X, with transfer function G(s).

$$G(s) = \frac{s^2 + 1}{s^4 - 6s^3 - 8s^2 + 86s + 87}$$

• *Implement System X in 20-sim using the linear system editor*
See the Frequency Domain Toolbox section in the 20-sim manual. You will have to design a controller for this system, but let's first look at the response of the system when it is not controlled.

• *Apply a step input to the (un-controlled) system. What happens to the output?*

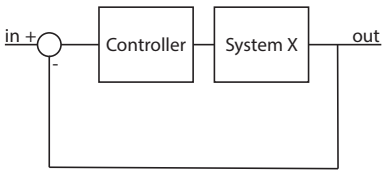

Figure 3.1: A standard feedback loop scheme

3.3 Initial Controller

To design a controller for this system several strategies are possible. First we try a proportional controller in a standard feedback loop, see figure Figure 3.1.

- *Implement a proportional feedback loop for the system*

To analyse this system you can use the model linearization toolbox, found under *Tools>Frequency Domain Toolbox>Model Linearization*. For more information on how to use this toolbox, see the 20-sim manual.

- *Use the toolbox to determine the closed and the open loop transfer function.*

To analyse the system you can use the root locus method. To generate a root locus using 20-sim, make a pole-zero plot inside the model linearisation toolbox. Then right click on the plot and enable the root-locus. Note that the root-locus of a transfer function (H) shows the location of the poles of the closed loop version ($\frac{k \cdot H}{1+k \cdot H}$) of that transfer function.

- *Generate the root locus of your system and controller.*

We would like to stabilize the system using only proportional gain.

- *Can you find a gain parameter that will result in a stable system using the root locus method? Why/why not?*

3.4 Extended Controller

We will now try to control the system using a controller with only one pole and one zero. Such a controller can be a PD, lead- or lag-compensator. For the control of our system we will focus on two controllers:

- A lead controller with a phase of approximately 55° at $5\,\mathrm{rad\,s^{-1}}$

- A lag controller with a phase at approximately -55° at $5\,\mathrm{rad\,s^{-1}}$

• *Use bode plots to find a lead and lag controller that meets these requirements.*

Implement a standard feedback loop with these controllers. Make sure you use the option in 20-sim that allows you to generate multiple implementations of a sub-model, see the 20-sim manual. You will now use the root locus to analyse the stability of the system.

• *What would need to happen with your root locus to stabilize the system?*

Use the linearisation toolbox to determine the transfer function of your system and plot the root locus of both controllers.

• *Which of these controllers can be used to produce a stable system?*

Now that we know that it is possible to stabilize the system using the root locus method, you need to select the gain of your system. You can use the probe in 20-sim to check what system gain is required to achieve certain pole locations. (Double click on the root locus line).

• *Use the root locus method and pick a gain which gives reasonable stability. Explain the criterion you used for this.*

Now you have found the gain that produces a reasonably stable system, it is time to apply this gain to the controller such that we can use it during further simulations.

• *Edit the gain of your controller such that it produces a stable system. What changed to the bode plot and the step response of your controller?*

Now it is time to verify that the controller that you made is actually stabilising the system.

• *Verify that your system now is stable by running a simulation and plotting output of the control loop.*

Congratulations you have stabilised the system! To analyse why this controller that we used resulted in a stable system we will take a look at the effect the controller has on the bode plot of the system.

• *Compare the bode plot of System X to the bode plot of the open loop gain of the stable system. Do you recognise the influence of your controller? What is the gain when the phase of the transfer function passes through -180°? Is the gain increasing or decreasing when the transfer function goes through -180°?*

3.5 Bonus question

The behaviour of the bode plot can be better understood using Nyquist diagrams.

• *Plot the Nyquist diagram of the open loop transfer function and the of System X itself. How many encirclements of the minus -1 point do you see? Are they clockwise or counter-clockwise?*

3.6 Integral term

The system now is stable however the final value is still different from the set point. We are missing an integral term.

• *Add the integral term to your controller, that is, add a pole at zero. Why does this increase the correspondence of the final value to the set point?*

The controller now is not stabilising the System X any more. It is your task to adjust the controller such that it stabilises the system again.

Note that the open loop transfer function simply consists of the multiple of both transfer functions. Therefore you don't have to use the linearisation toolbox, but you can also simply add the poles and zeros of your controller to those of System X. This allows you to quickly determine the response of the system.

Tuning the controller such that it again is stable can be quite tedious. To do this more systematically you could use the root locus to optimize a specific parameter. For example the location of a specific pole or zero. To do this you will have to solve the root locus equation for that specific parameter.

Remember the root locus of a certain transfer function H(s), shows the location of the poles of the closed loop version of this transfer function:

$$\frac{k \cdot H}{1 + k \cdot H}$$

This is equivalent to saying that in order to calculate the root-locus you have to calculate the solution of:

$$H = -\frac{1}{k}$$

.

To calculate the root locus for another parameter, e.g. the effect of adding another zero:

$$G(s) = (s + z)H(s)$$

You have to solve the root locus equation described above for p e.g.

$$z = \frac{-1 - skH(s)}{kH(s)}$$

Next invert and negate it to obtain:

$$-\frac{1}{z} = \frac{kH(s)}{1 + ksH(s)}$$

When 20-sim is asked to calculate the root-locus of the right part of this equation it will still show the location of the poles of the closed loop transfer function, but this time as a function of the zero z instead of the gain.

- *Adjust the controller such that it is stable with the integrator. Explain why you add or move each pole or zero, using root locus or bode diagrams. Optimise at least one parameter other than the gain using the root locus method.*

Hint: You can use the model linearisation toolbox in 20-sim to calculate your transfer function!

Hint: It is not possible in 20-sim to have a transfer function with more poles as zero, however there is possible to compute a pure derivative using the derivative block.

3.7 What to show

You do not have to show all your answers/model to the SAs to complete this pre-project. You should show the following to your SA:

- Your conclusions about why (or not) the system can be stabilized with proportional control.
- The design of the controller according to the requirements in section 3.4.
- The bode plot of open loop gain of the controller you stabilised System X with .
- The poles and zeros added or moved in order to stabilise the controller with the integral term, including the reason why.
- One root-locus plot with a parameter other than the gain of the system.

Remember that your written journal is your proof of work. If it is not written down, you did not do it.

Chapter 4

Pre-project 3: Characterization

4.1 Introduction

The goal of this pre-project is learning how to estimate parameters of a system based on matching between a model and measurement data. This is done by determining the parameters of a motor model, based from purely electrical measurements.

In this manual you will find a series of questions/commands. You do not have to hand these in, but you should note the answers down in your journal. When asking questions, you should be able to show these answers.

To complete this pre-project you need to show a SA your work, so that they can sign off on it. To sign-off you have to add your group number to the queuing list, see section subsection 1.1.4. What you exactly need to show is defined in section 4.6.

In any of the simulation files you get during this tutorial, the current is in amperes, while the voltage is in volts.

4.2 Initial Model

Before you can determine the model parameters you will need to create a model. The system that you are modelling is an electric motor with different wheels attached to the axle as a load. The load will change during the experiment. Thus it is important to keep the motor and the load separate.

The load can be modelled as only a moment of inertia. For the motor it is useful to include at least the coil inductance and resistance, and the moment of inertia and the friction of the rotor. You may assume the motor axle to be rigid. The motor is controlled with a voltage.

- *Create a motor model based on this information.*

4.3 Experiments

To determine the motor parameters, a series of experiments have been performed on the motor setup. Below you will find a description of the experiment, and a reference to a datafile that can be imported into 20-sim. To read imported data you can use the *DataFromFile* component, found under: *Library\Signal\Sources\DataFromFile*.
When loading measurements, make sure your simulation time step is small enough to actually see the data! Depending on the integration method this can be done by either setting the step size or the maximum step size to a small enough value. This setting can be found in the run properties pop-up menu, under the tab of the used integration method.

For every experiment try to answer the following questions:

- *What is the exact situation of the experiment, as represented in your model?*
- *What information can you get from the experiment?*
- *Is this actual new information?*
- *How can you use this to determine model parameters?*

Based on this information you determine your model parameters. Furthermore an experiment description may contain additional question. The experiments start with a motor without load.

Experiment 1

For the first measurement, a lever is connected to axle of the motor. The other end of this lever is resting on a scale see figure Figure 4.1. The horizontal distance between the center of the axis of the motor and the point where the lever touches the scale is 24 cm. A voltage pulse is applied to the motor. This pulse lasts 5 seconds and has an amplitude of 12 V. The result of this pulse can be found in *measurement1.csv*. The weight measured by the scale is 23 g. The motor is rotating in the direction of the scale and the scale was reset such that the original weight of the lever was not taken into account.

The resulting graph has exponential behaviour. Fitting data which has exponential behaviour becomes easier when the data is plotted with a logarithmic y-axes. This is because of the following mathematical identity:

$$\log(a \cdot e^{-b \cdot x}) = \log(a) - b \cdot x \tag{4.1}$$

Where log is the natural logarithm. Because of this identity a plot of the logarithm of an exponent it will result in a line, which it way easier

Figure 4.1: The setup used during experiment 1.

to fit. To calculate the logarithm in 20-sim use the Function-log block found under *Library →Signal→Block Diagram Non-linear*. Because a logarithm is only defined for positive values you to first make sure the data is positive by passing it through a Function-Absolute block, also found under *Library→Signal →Block Diagram Non-linear*.

- *Determine the coefficients of the exponential decay in the current when the voltage is reduced back to zero.*

- *How would you model a clamped axle in your model? What effect does this have?*

Experiment 2

For the second measurement, a disk is attached to the motor axle.
This disk is described by the following text:
The disk is laser-cut from Delrin (density $1.41\,\mathrm{g\,cm^{-3}}$*). The disk has a thickness of* 6.2 mm*. The radius of the disk is* 43 mm*. The hole for the axle in the disk has a diameter of* 8.2 mm*. The disk is solid and the lasercutter can be assumed perfect.*

A sinusoidal input voltage of 12 V at $1\,\mathrm{rad\,s^{-1}}$ is applied to the motors. The data resulting from this pulse can be found in *measurement2.csv*

- *What is the expected moment of inertia of this disk?*

Experiment 3

For the third measurement the same disk is attached to the motor axle. However this time the applied input signal is the same pulse as in experiment 1. The resulting data can be found in *measurement3.csv*

4.4 Extra Sensor

With the previous experiments you should have determined all the model parameters. To validate these model parameters, an extra experiment has been done. For this validation an angular velocity sensor is connected to the motor axle. The data from this validation experiment can be found in *validation.csv*.

- *Does your model correspond to the validation data? If not, fix it.*

4.5 The real wheel

To also model the wheel attached to the motor, the wheel is attached to the axle, all other disks are removed. The result of this measurement can be found in *measurement4.csv*

- *Can you now determine the moment of inertia for the wheel? How?*

4.6 What to show

At the end of this pre-project you have to show the following to your SA:

- A bond graph model containing your motor and the wheel.
- A table of all determined parameters.
- Details on what experiments you used to determine each parameter, and a short explanation. For example: "R_{motor} was determined by dividing the steady state voltage by the current in experiment 1."

Remember that your written journal is your proof of work. If it is not written down, you did not do it.

Chapter 5

Pre-project 4: MISO control

5.1 Introduction

During this pre-project you will design a controller for an inverted pendulum.

In this manual you will find a series of questions/commands. You do not have to hand these in, but you should note the answers down in your journal. When asking questions, you should be able to show these answers.

To complete this pre-project you need to show a SA your work, so that they can sign off on it. What you exactly need to show is defined in section 5.9.

5.2 MATLAB connection

During this pre-project you will use MATLAB in combination with 20-sim.

5.2.1 Windows

On windows the MATLAB connection of 20-sim allows you to easily export transfer functions from 20-sim to and from MATLAB. To make this connection work on Windows 10 you will need to:

- Make sure you have MATLAB including the Control System Toolbox installed
- Add the folder that contains the file "libeng.dll" to your path variable. On most systems this will be something like `C:\ProgramFiles\MATLAB\R2019b\bin\win64`. To add a folder to your path variable:

  Search for "edit the system environment variables" and open the first hit

Click on Environment Variables...

Select the path variable and click on "Edit..."

Click on new and add the folder that contains "libeng.dll"

- Register MATLAB as an Automation server. This can be done by running MATLAB as Administrator and running the "regmatlabserver" command inside the MATLAB command window.

To export a transfer function from 20-sim to MATLAB from the linear system editor in 20-sim, click on the button with the MATLAB logo and the arrow to the right. 20-sim will ask you to name the transfer function. This name will be the name of the transfer function object in the MATLAB command window that will open as soon as you press OK.

To import a transfer function from MATLAB to 20-sim from the linear system editor in 20-sim, click on the button with the MATLAB logo and the arrow to the left. 20-sim will ask you the name of the transfer function you want to import. This name should be the name of the transfer function object in the MATLAB command window that was opened previously.

5.2.2 Other operating systems

When running Matlab on a different operating system as windows, 20-sim cannot establish a connection to MATLAB. However transfer functions can still be imported by manually copying the coefficients of the transfer function. To generate a new transfer function in MATLAB using these coefficients use MATLAB's tf function. To import for example the transfer function $\frac{s+1}{2s+3}$ run:

```
G = tf([1 1],[2  3])
```

To copy a transfer function from Matlab to 20-sim use the following commands to output the transfer function in a format that can easily be copied.

```
tf(C).numerator{1}
tf(C).denominator{1}
```

Where C is the transfer function that you want to copy.

5.3 MATLAB sisotool

During this pre-project you will use MATLAB's control system designer. After importing the transfer function of your plant, run the following command to start the control system designer.

```
sisotool(G)
```

Where G is the name that you gave to the transfer function of your plant. Inside the control system designer, it is possible to add poles and zeros to your bode plot or root locus. This can be done by first selecting the corresponding plot by clicking on it and then selecting either a pole or a zero in the root locus editor / bode editor tab. The gain of the system can be edited by dragging the location of the poles in the bode plot or dragging the bode plot up or down.

After you have edited the transfer function such that it as desired, export the transfer function from the control system designer to the MATLAB workspace by clicking export and then selecting the C parameter.

5.4 System

During this tutorial you will design a controller for an inverted pendulum, a model of this system can be found in `model.emx`. The motor of the pendulum is controlled using a motor controller that contains a current sensor. The angle of the pendulum is measured using an encoder and the rotation speed is measured using a gyro, which both have a limited bandwidth and contain noise.

5.5 Rotation control loop

We will first cheat a bit and design a controller using rotation output of the model, which gives us the rotation output without bandwidth limitations and noise.

- *Find the relevant transfer function in 20-sim and use MATLAB's control system designer to design a control loop that is stable and able to rotate the pendulum from* 0 rad *to* 0.1 rad *and back.*

5.6 Sensor fusion

The two sensors on the pendulum, both have their strength and weaknesses. The encoder directly measures the position of the carriage, but it's bandwidth is limited to 10 Hz. The gyro has a higher bandwidth, however to obtain a position, it's output has to be integrated. This makes the position output of the gyro sensitive to low frequent drift.

To get to a reliable and high frequent reading, the sensors outputs need to be combined. To combine these sensor output one could for example use a so called complementary filter, which is nothing more as two filters with a transfer function with a sum of one.

- *Combine the output of the encoder and the accelerometer to obtain one high frequency position output. The output of the filter can be checked*

using the "rotation" signal.

- *Check if the control loop that you designed during the previous assignment is able to stabilise the system with the combined output.*

5.7 Current control

To have better control over the electrical power dissipation inside the motor, we would like to control the motor using current control.

- *Design a control loop that controls the current through the motor.*

- *What does it mean that in the closed loop transfer function there is an unstable pole on top of an unstable zero?*

- *Design a rotation control loop that is in cascade with the current control loop and that is able to stay balanced with a step input with an amplitude of* 0.1 rad.

Hint:You can use the flow of the MSe to get the transfer function from the current to the rotation without the poles and zeros of the current control loop.

5.8 Digital system

In practice often a digital controller is used, this often introduces delays into the system, for example in the analog to digital and digital to analog conversion. The 20-sim model `model_with_delay.emx` contains a model of the same pendulum, however this time a model (forward Euler) for a delay of 1 ms is added for each ADC and DAC.

- *Is the current control loop that you designed still stable?*

- *Design a rotation control loop that is able to stay balanced with a step input with an amplitude of* 0.1 rad. *(without a current control loop, with sensor fusion)*

Hint: You can use the "rotation_delayed" signal to get a transfer function which includes the delays, but without the poles and zeros of your fusion filters.

5.9 What to show

At the end of this pre-project you have to show the following to your SA:

- Your rotation control loop using the rotation signal.
- The filters used to combine the encoder and the gyro signals.
- Your rotation control loop using the sensor fusion output.
- The rotation control loop that includes the current control loop.
- Your rotation control loop for the system with delays.

Remember that your written journal is your proof of work. If it is not written down, you did not do it.

Chapter 6

Project 1: Motor measurement

6.1 Introduction

Before you get started on modeling and controlling the segway, you will start with a bit of a smaller project, more specifically the motors of the segway. In order to let the segway drive around it is essential to be able to accurately control the motors.

The final goal of this lab is to make one wheel follow the position of the other wheel. In order to be able to design this controller you will create a model of one of the motors. This goal is represented by the assignments found in this lab.

As the goal is not only to be able to rotate the wheel, but also to make a correct model and controller, specific attention will be spent on matching the model characteristics to that of the motor, to compare the motor model with the real behavior and not least, to compare the performance of the controller on the model and the real motor. These steps will be explained in the following parts.

6.1.1 Grading

As is mentioned in the introduction, this lab will account for 30% of your project grade. This grade is based on a short (4 pages) report together with the simulation files. More specifics on what to hand in can be found in section 6.5.

For more information on how the reports are graded see Appendix C.

You should also keep a journal to be able to show what you have done and measured.

6.1.2 Segway

A segway will be assigned to your group. Due to limitations, you will have limited time to test on the actual segway. You will have to register

for measurement slots. This means you will have to create a model to work with, in order to test your controller.

For more information on how the segway functions, see Appendix A.

6.2 Motor model

Before characterization and control of the motor you have to start with a model. The model of the motor is first made in a general sense. You will identify behaviors present in the motor and how to describe them. Then you take a look at the environment of the motor. At the end of this section you have a model of the motor, including the environment that the motor is in.

To guide you through the modelling process we have outlined a few questions to answer and steps to take. Take note, you do not have to hand in these questions in your final report. However, when you ask for help it is good to have these answers present. So write them down!

The exact specifications of what to hand in are given later in this manual. (section 6.5).

Model components

To determine the components present in the motor, you should first identify the different domains present in the motor, or at least the domains present on the outside of the motor. For all of these domains look at the components/behaviors present, and how to model them. Also identify the component between the domains.

- *Make an overview of these components.*

Significant behavior

When using a model for a problem, the model should be a simple as possible, and as complex as required. Thus you need to evaluate all the components to see if contribute to the behavior you are interested in. For now you use the motor only for a controller that moves a wheel or load to a certain rotation.

- *Evaluate your components in your overview on how significant they could be for the current lab.*

Model Environment

The motors will not be used in a vacuum, in both the literal and the figurative sense. So you need to model the motor environment. If you look at the motor you will see there are 2 ports with the environment. On one side you have the electrical port, on the other side the motor axis.

To predict the motor behavior accurately, the environment attached to these ports need to be modelled.

The electrical side of the motor is connected to the controller of the segway, with a system to turn the controller signals into voltages for the motor. You can look up information of the segway in Appendix A.

The mechanical side can be connected to a lot of things. For the first model assume that the motor is connected a wheel.

- *Model the motor environment, are all elements still significant?*

Best practices

When creating your model it is important to conform to the best practices as shown in the 20-sim manual. So make sure your model is annotated with the different domains, that the units are correct and that all parts have the correct names. You don't need to put the different parts of the model in different sub-blocks. But you should clearly show the separation between the motor, the electrical environment and the mechanical environment. In short, make your model as easy to understand as possible in a single view.

What to report

From this section you will need to report two things. First your finished motor model. Second, the decisions to or to not, neglect certain model components.

6.3 Motor characterization

Now that there is a model for the motor, you need to determine the parameters of the motor. To determine these parameters you will need to perform a few experiments.

- *Make a list of all the parameters that you need to determine.*

To determine the parameters of the motor, you can use the tools available with the segway. The specifics of these tools are available in section A.5. When using these tools, take into account a few things:

- You do not need to use all the tools that are available in the kit.
- You could also influence the wheel using your hands, do not do this when the wheel is moving quickly!!
- Spring balances are orientation dependent, make sure you calibrate it for the orientation it is used in.
- A rope with weights is limited in how far they can move, make sure to keep your experiments within these limits.

- When using the current sensor, make sure you are using it in a region where it is linear (which it is when the absolute current is larger as 0.3 A)

To execute models on the segways connect to the segway using 20-sim 4C. More information on this can be found in Appendix A and in the 20-sim manual.

• *Design and perform a set of experiments to determine your parameters, make sure that you cover all the parameters in your model.*

Parameter reflection

After you have determined the parameters of your model, you should reflect on the values of the parameters. Is there a way for you to determine if the results are realistic? How big/small would a component with these values be?

• *Reflect on your parameters.*

Furthermore your values might not be completely stable. For example the resistance of the motor coil, is very dependent on the temperature of the motor. Have you noticed this effect?

• *Do you need to take this effect into account for your model? Why/why not?*

Parameter validation

To validate the performance of the model, compare the model to the real motor.

• *Apply a appropriate test signal to your motor and look at the: angle, velocity and rotation of the motor.*

In order to compare these results to the real motor you will have to connect to the real segway. When sending this signal to the segway, take into account the following:

- The signal output from the controller is not in volts, you will have to convert the test signal.
- The angle provided by the encoder will loop back to 0. You will need to convert this back. You can use the *encoder_to_total* block provided on canvas.

- The output of the angle sensor is in rotations, not in radians.
- The segway controller has a discrete timestep, make sure this is comparable to your model.

For more information on the segway hardware, see Appendix A.

- *Compare your model and the real motor using the test signal.*

When comparing the motor and the model, look for the differences, can you explain the differences and will they be relevant for controlling the segway? You can also correct the model using this test signal. A slight tuning of the model parameters might result in a better fit. You can use the parameter sweep/curve fitting feature of 20-sim.

- *Correct your model using the real life results, note down the corrections used.*

What to report

From this section you need to report a list of the experiments performed, with a short description, and what parameters where determined with the experiment. Furthermore you need to show a table with the final motor parameters. You also need to hand in your validation simulation. To hand in your simulation hand in a 20-sim file that when opened can immediately be simulated and will show a correctly formatted validation. Refer to the 20-sim guide for formatting.

6.4 Control

The final part of this assignment is to design a controller that makes the rotation of one wheel follow the rotation of the other wheel. First you will design a controller in 20-sim, then you will test it on the real motor. For this test you may assume a wheel to be connected to the motor. You will design the controller based on a transfer function of me model. You can generate this transfer function using the linearization toolbox in 20-sim. For more information see the 20-sim guide.

- *Generate a transfer function that can be used to design your controller. Don't forget that the PWM signal is not volts and the angle sensor output is not in radians.*

Controller design

You need to design the controller based on the transfer function. You can either add poles and zeros to the transfer function in 20-sim or use the MATLAB control toolbox.

- *Design a controller based on the transfer function.*

Implement your controller in 20-sim, make sure you keep your controller separate from the other parts of your model such as the control electronics, the motor, and the load. You can use the linear system editor or a dedicated controller block. Also take into account the fact that the real segway has a conversion from signal to voltage in the electronics and that the output of the angle sensor is in rotations instead of radians.

- *Implement a controller, describe how.*

When implemented you can test your controller on your model. Based on this performance you could tune your controller slightly.

- *Evaluate and possibly tune your controller.*

Controller realization

Take into account that the sensor signal wraps back around to zero, and that the sensor might start at an unknown value. To unwrap the signal there is a 20-sim model called encoder_to_total.emx available on canvas. For more information on the segway sensors see Appendix A.

You can now run the controller on the real motor. You can then compare the performance of the controller on the motor and compare it to the model. Which one follows rotation signals better? What differences can you see?

- *Test your controller on the real segway and compare it to the performance on your model.*

What to report

For this section you have to hand in your calculated controller with an explanation on how this was done, and the decisions you made to tune your controller, so what changes were made and why. Furthermore hand in a 20-sim file which shows the comparison between the controller performance on the model and the real motor. Make sure that when opened this file can immediately be simulated and will show a correctly formatted comparison. Refer to the 20-sim guide for formatting.

6.5 Hand in

To summarize you have to hand in a short report of about 4 pages and some 20-sim files. This report will have to contain:

- The model design

  - Your finished motor model, including an explanation.
  - The decisions to or to not, neglect certain model components.

- The parameter characterization
  - A List of the experiments performed, with description and parameters estimated.
  - A table of the final model parameters.
  - The controller validation.

- The controller
  - Your calculations on the controller, including explanation.
  - Possible tuning decisions made.
  - A comparison between the real controller and the simulated controller.

Refer to the specific sections for more detail. The report file will have to be handed in as **.PDF** and should be named:
groupXX_motor_lab.pdf.
The same goes for the 20-sim files, name them:
groupXX_motor_validation.emz and **groupXX_controller_validation.emz**.
When handing in 20-sim files that contain a data from file block, please save them in the emz format. Please do not hand your data as an archive (i.e. as a zip). For more information on how the reports are graded see Appendix C.

Chapter 7

Project 2: Segway lab

7.1 introduction

The final and biggest part of the project is to model and control the Segway. The first week will be spent on modeling, the second week on control. The segway model has to be shown to your supervisors in a short presentation. To test your control of the Segway, a set of challenges is defined. For more information on these challenges see subsection 7.1.4.
To create your model and controller there is some information and a series of questions has been written down in this manual. You do not have to hand in all the answers to all these questions, but it is advised to write them all down in your journal.

7.1.1 Segway model presentation

Halfway through the project, on Monday 20th of January, you have to present your Segway model. This is done by a short pitch for a set of Supervisors. This pitch takes about 10 minutes and you will show of your model and will have to answer questions about this. You will also get feedback on your model which you can use for the rest of your project.

For this pitch it is important that:

- Your segway model is properly formatted (see the 20-sim guide).
- You have a working simulation of your segway falling over.
- You can visualize the position/angle of the segway.

7.1.2 Grading

As is mentioned in the introduction, this part of the project will account for 70% of your project grade. This grade is based on the report together with the simulation files. More specifics on what to hand in can be found in section 7.9
Bonus points can be earned during the challenges. The best scoring

group will earn half a point, the second group 0.4, the third group 0.3, etc.
For more information on how the reports are graded see Appendix C

7.1.3 Segway

A segway will be assigned to your group. Due to limitations, you will have limited time to test on the actual segway. Your group has been assigned measurement slots on Canvas. This means you will have to create a model to work with, in order to test your controller.

For more information on how the segway functions, see Appendix A.

7.1.4 Challenges

You will have to complete 3 challenges:

1. Standing still with the segway.
2. Standing still with a disturbance (either inclined floor or a push).
3. Driving from start to a finish, turn around, and go back.

The last two challenges will be done on our test track: it is 5m long, it has a starting area and a finish area and a line along the full length of the track. Each challange has to be completed within the time limit (will be defined later). These challenges will take place during the last day (24th of January).

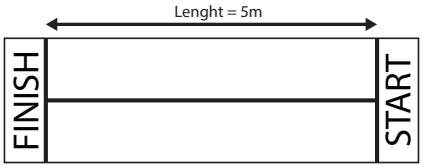

Figure 7.1: An overview of the test track

We encourage you to do these challenges autonomously, using the line camera underneath the segway, but it is allowed to use the gamepad.

In case your segway has passed the challenges, you qualified for the segway soccer tournament. In a segway soccer match your segway will battle against another segway. The goal of segway soccer is to push a beach ball into your opponents goal. Segway soccer is played on the field described in figure 7.2.

7.1.5 Hints and questions

The following sections of this assignment will provide you with a set of subjects to think about, including some information, some hints and some questions. These questions do not need to be answered directly in your report. But you should note the answers and thoughts about these subjects down in your Journal.

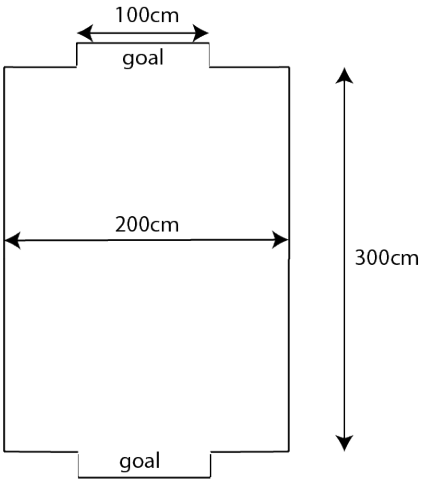

Figure 7.2: Dimensions of the segway soccer field

7.2 Segway Model

When creating the model of the segway, you should take into account a few things. First and foremost is that the motors provide their torque between the body of the segway and the wheels. Second the wheels can be assumed to be fixed in the vertical direction (contact with the floor).

When making the model you will need to make sure that all the signals that are measured by the sensors are present in the model, as you might need them for your controller later. You might need to construct extra points on the body of the segway for this.

When linearising the segway model for controller design, you should linearize this without all the complicated sensor effects. This allows for a simpler controller design. The multiple implementations of sub models feature, could help with this.

As a full 3D motion is to much for this course. Fortunately, you can model the segway as a 2D rigid body. This model can then be made in the Forward-Up-Plane.

7.3 Angle sensor

There is no sensor that can directly measure the angle of the segway body in relation to the world.

To determine this angle you can use both parts of the IMU. The IMU contains an accelerometer that measures the direction of gravity and a gyroscope which measures angular velocity.

Both of these sensors have their advantages and disadvantages.

- *Determine the advantages and disadvantages of using the gyroscope or the accelerometer.*

These disadvantages can be offset using clever tricks or by combining

both sensors. For example by intelligently setting the begin value of the gyro. You can also try to combine both sensors. In that case it is advised to look up matched filtering.

The most important part of your angle sensor design is that you verify the angle measured by the sensors on your segway.

- *Do an experiment to validate the performance of your angle sensor implementation.*

Is your sensor good enough to use?

7.4 Motor Control

The motors on the segway are controlled by voltage, however you could choose to control the motors differently based on a local control loop.

The three options that you can use to control your motor are limited by the sensors and are:

- Voltage
- Current
- Velocity

Depending on which of these inputs you use, your segway will behave differently.

- *What is the effect of the different ways to control the motor on the transfer function of your model*

- *Which of the control methods would you want to use, and why?*

- *How can you implement this control method, and what are the limitations?*

7.5 Multiple inputs

For the challenges you will have to control multiple aspects of the segway. Not only the angle, but also the velocity and the position need to be controlled. However you only have a single output to control the segway.

You will need to find a way to combine all the different inputs/setpoints for your controller. We have three options for you on how to do this, but feel free to design your own.

The first option is to cascade your controllers, the position controller determines the setpoint for the velocity controller, which determines the input for the angle controller.

- *What are the advantages and disadvantages of this method? What can you say about the speed of the different controllers?*

The second option is to place the controllers in parallel, you would need to find a place in your system to combine the outputs of the controllers together.

- *What happens when these controllers disagree over what should happen? Is this a problem?*

- *What would be a good place to add the different signals and why?*

- *Can you make one controller more important than the other controllers? Do you want to?*

The third option would be to combine your error functions before sending it into the controller. This would allow you to use one controller.

- *Are all error of the same importance? How would you correct for this?*

- *Are there disadvantages to using this method?*

Whatever way you pick or design. Think about the limitations and advantages, and don't forget to validate your assumptions.

7.6 Direction

The direction of the segway is determined by the difference in wheel velocities, which is a signal you can measure.

- *Can you also control this signal?*

- *How would you approach this?*

- *Can you split the different wheel velocities into a forward and rotational velocity?*

7.7 Challenges and control

For the challenges and the segway soccer tournament, you have to be able to control the segway using the gamepad.

- *Where in your control loop will you insert the control signal?)*

• *What is the maximum control signal you can give before the segway becomes unstable?*

7.8 Safety

When working with the segway, it might fall over. It is a good idea to stop the segway when it has fallen. It might even be possible to let it restart when you put it upright again.

• *How can you detect that you have fallen over?*

• *How can you shut off the motors when you have fallen over? Look into switches.*

When restarting the segway you might have to compensate for integrators in your controller that have wound up to large values.

• *Find a way to prevent windup or to reset your intergrators.*

7.9 Hand in

To summarize you have to hand in a report of a maximum of 16 pages (excluding your appendix) and some 20-sim files. This report will have to contain, but is not limited:

- An overview of the segway model and characterizations
- Decisions made during the modeling process
- Decisions on what kind of controller to make
- The design of the controller.
- Validation of the segway model, including sensors and parameters
- The performance of the real segway.
- The performance during the challenges.

See these specific hints sections for more content to show. Also hand in the final model. The report file will have to be handed in as **.PDF** and should be named:
groupXX_segway_project.pdf.
. The same goes for the 20-sim files, name them:
groupXX_segway_project.emz. When using data that is loaded from a file, please save your 20-sim file as an emz. Also please do not hand

your data as an archive (i.e. as a zip), this prevents us from using the speedgrader in canvas. For more information on how the reports are graded see Appendix C

Appendix A

Segway Specifications

The segway used for these labs is a custom made platform based on a raspberry pi 3B. An image of this segway is shown in Figure A.1. This segway contains 2 motors, with a current and angle sensors, a IMU and a line sensor.

All of these motors and sensors are connected to the controller in such a way that they can be used in 20-sim 4C. How the signals is 20-sim4C translate to the real world, is described in the specific sections. The segways are accessible by connecting to *ramsegwayXX.roaming.utwente.nl* where XX is the number of the segway. The frequency at which the controller on the Segway can run depends on the model, but is approximately 1 kHz

With the segway you are supplied a box of measurement instruments, this box contains a measurement disk, a spring balance, and various tools to perform measurements on the segway.

A.1 Motors

The segway contains to brushed DC motors, controlled by two MC33926PNB motor driver chips. These motor drivers are powered by 24 V and controlled with a PWM signal. The signals from the controller are mapped from -1 to 1 for a duty cycle of -100% to 100%. Thus resulting in a effective mapping from -1 to 1, to -24 V to 24 V.

For example, a signal of 0.4 will result in a duty cycle of 40% and thus an effective voltage of 9.6 V.

Take note that both motors are mirrored, thus the motors will turn in opposite direction (forward and backwards) when the same signal is applied.

Every motor has 2 sensors, a current sensor and a angle sensor.

A.1.1 Current sensor

The motor driver outputs a current signal. This current signal is sampled by a AD7091R-5 and sent to the controller. The current sensor can only sense the magnitude of the current, not the way the current is

Figure A.1: The segway used for these labs.

flowing. The bandwidth of the currents sensor is also limited to 80 Hz and the current sensor does not measure currents below 0.2 A and is not linear below 0.3 A, see figure Figure A.2. After 0.3 A the current (I) can be calculated from the current sensor output (O) using the following formula:

$$I = 4.7 \cdot O - 0.09 \tag{A.1}$$

For example, both a current of -1.7 A and 1.7 A, result in the signal 0.38 in the controller. Also both a current of 0.05 A and a current of 0.1 A will result in a an output of 0. When using this sensor for parameter identification, make sure you it in it's linear range. For example use a sine with and offset instead of just a sine.

A.1.2 Angle sensor

There is a magnet connected to the back of the motor axle. The orientations of this magnet is measured with an AS5048B angle sensor, which

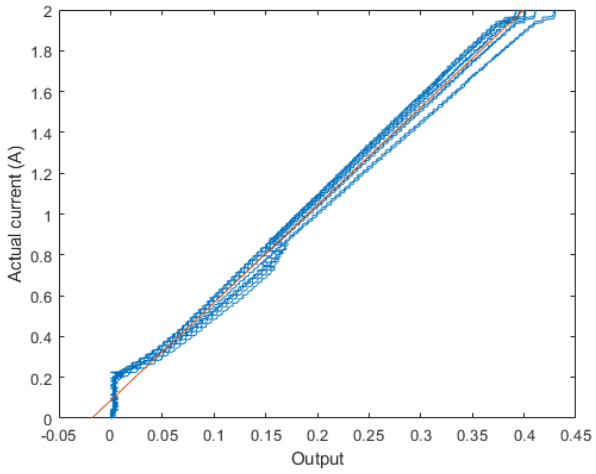

Figure A.2: The actual current versus the output of the current sensor.

measure the direction of the magnetic field. This means that after a complete revolution of the motor, the angle sensor will output the same value.

The angle of the motor is mapped in such a way that the angle of 0 till 2 π corresponds to a signal from 0 to 1. The sample frequency of the angle sensor is equal to the frequency at which the controller is running, but has one sample delay.

For example if the motor is at the angle π (signal 0.5) and turns a quarter turn the resulting signal will be 0.25 or 0.75 depending on the direction of turning.

A.2 IMU

The segway can try to measure its orientation by using an Inertial Measurement Unit (IMU), more specifically the BMX055. This chip can measure both acceleration and angular velocity. The chip is connected to the controller via a digital interface in such a way that:

Acceleration in the X, Y and Z directions is mapped from -2 to 2 g to a signal of -1 to 1. This means that the acceleration felt due to gravity is mapped from $9.81\,\mathrm{m\,s^{-2}}$ to a signal of 0.5.

Rotational velocity around the X, Y and Z axis is mapped from $-500\,^{\circ}\mathrm{s}^{-1}$ to $500\,^{\circ}\mathrm{s}^{-1}$ to -1 to 1. For example, a rotation around the Z axis of 50 degrees per second, will result in a signal of 0.1.

The sample frequency of the both accelerometer and gyro is equal to the frequency at which the controller is running, but has one sample delay.

A.3 Line sensor

The line sensor is based on a camera at the bottom of the segway. This camera is connected to a bit of image processing software to determine the line location. Based on the image the camera sees it will provide the following information on the position and angle of the line seen.

This information is presented as 4 points. These 4 points are the locations of the left and right side of the line, on the top and the bottom of the camera view. Here a signal of 0 means completely on the left of the screen, and 1 means completely to the right. For details on the names of the points, see Figure A.3

It also presents the color of the line seen, expressed as the angle in the YUV plane, mapped from 0-1. It is advised to calibrate these color values.

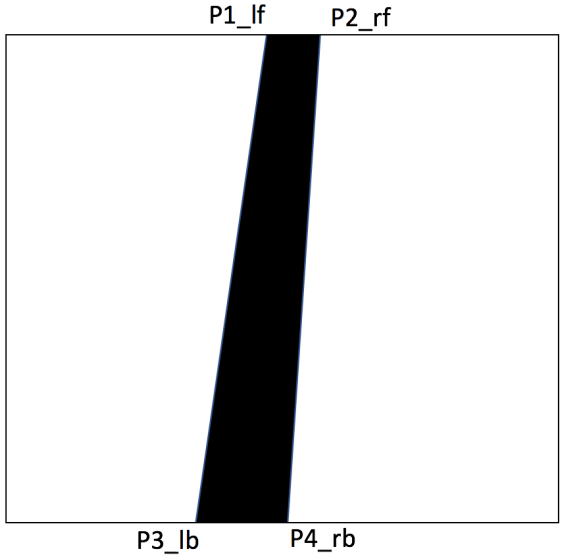

Figure A.3: The crossover points of the linesensor.

A.4 Gamepad

The segway is paired with a bluetooth gamepad. The 20-sim target is configured to read the buttons and analog channels of this gamepad. The buttons are either 0 or 1, and the analog channels go from -1 to 1. For the specific positions of the different buttons/channels, refer to Figure A.4.

Figure A.4: The gamepad provided with the Segway

A.5 Measurement instruments

There is a measurement disk provided with the segway. This disk has a set of hooks to attach spring balances or weights, and can be connected to the motor axle. A diagram of the disk is found in fig A.5. The disk has a moment of inertia of approximate $1380\,\mathrm{g\,cm^2}$. And a mass of 119,6 grams.

Bolts can be added to the disk to increase the moment of inertia. The bolts and nuts weigh 16.5 g to 4.6 g respectively.

Furthermore a spring balance, a set of weights (50 g each), rope and a wrench are provided. Pay close attention to the scale on the spring balance, some of them are marked in newton, some of them in kilogram.

Make sure to tighten the wing-nut when the measurement disk is attached.

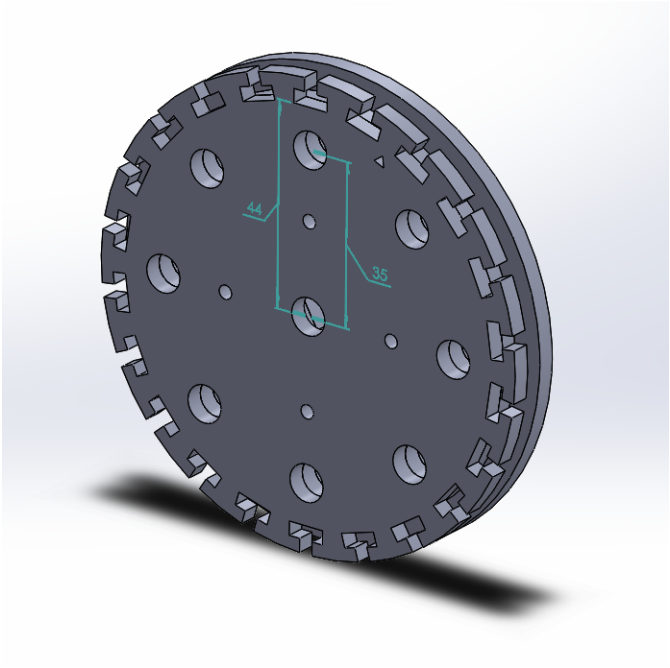

Figure A.5: Overview of the dimensions of the measurement disk (in mm)

Appendix B

Segway troubleshooting

There are two possible errors that can be generated by the 20-sim 4C connection. How to fix these errors is described in this section. Furthermore this section contains some information on the batteries of the segway and restarting it.

B.1 Possible errors

This section contains of lists that you may encounter during your experiments as well as the solutions to these errors.

Error: Target could not be initialized
Solution: Make sure your segway is on and you have and that you are connected to eduram. If this is the case this error is likely caused by the fact that 4C cannot find the segway device and does not retry the connection. To fix this, make sure your hostname is correct in the target section of 4C. Save your project and restart 4C.

Error: ipcv library could not be loaded
Solution: This error is usually caused by connecting a new instance of 4C without properly closing the previous connections. To fix this, restart the segway, see section B.2.

Error: tokenparser.xml not found
Solution: You unzipped your target in the wrong directory. The folder C:/program files (x86)/20 sim 4C/target/ should contain a folder a called raspberry-pi-linux.

Error: 20-sim-4c does not have a target called Raspberry Pi Linux.
Solution: Make sure you unzipped the target that can be downloaded from canvas into the folder C:/program files (x86)/20 sim 4C/target/

Error: 20 sim-asks for a license
Solution: Click on activate and use the license code that is supplied

via the wiki.

Error: I plugged in the charger of my segway but the led on the power shield stays green (not charging).
Solution: Turn on the segway.

Error: error 14: ivcCommand.getValuesBinary –¿ Application is not running..
Solution: Turn on the segway.

B.2 Restarting the segway

To restart the Segway follow the following steps:

1. Make sure the adapter is disconnected.
2. Hold the button on the segway until the red light on battery back blinks and the lights turn off.
3. Press the button again.

Note that when the segway is starting up, it cannot be shutdown. To shut it down you have to wait for about 30 s.

B.3 Batteries

The segway has a set of LEDs on the motor shield on top of raspberry pi. The colors of the lights have certain meanings in relation to the batteries:

- **BLUE** the batteries are fully charged
- **RED** the batteries are charging.
- **GREEN** the segway is on but isn't charging.

When none of these light are on, the segway is off. The segway should be able to run for about **30 minutes** on a full charge.

There also is a set of LEDs on the battery pack inside the segway. The LEDs on the battery pack will blink **RED** on both start-up and shutdown. Besides that the colors of the lights have the following meaning

- **GREEN** The battery pack is on and functioning normally
- **RED** The batteries are too full.
- **YELLOW** The batteries are too empty.

Note that the green LED is very dim.

Appendix C

Report Criteria

You have to hand in a report for both the projects. For the report we provided you with a template. You can choose between Word and LaTeX.

C.1 General Report Content

To make it easy for people to understand your report it is good practice to follow a standard report structure. In this case you could follow the following structure:

Introduction describe what the reader can expect while reading your report and what you are trying to achieve.

Analysis explain what you expect and what information you require to do the measurements.

Methodology contains what you are going to measure and how you are going to measure it.

Results show the outcome of your (processed) outcome of the measurements. It is not required to proof your measurement. However, it is important that the reader can reproduce your results.

Conclusion finish up your report. Reflect on your introduction and discuss if you succeeded or not.

Discussion Discuss what went wrong, why it went wrong and what can be improved for future work.

The final form of this basis can be changed. However, make sure that the content of this list is still in your report.

C.2 Checklist Report

Any scientific report has to meet a list of requirements, which make sure that your report can only be interpreted in one way.

- Did you use the correct units for each value?
- Does every figure have a caption?
- Does every figure have units on the x and y axis?
- Can the axis of each figure be read without zooming in? If not rescale your 20-sim simulation window such that is not full screen anymore.
- Can your diagrams be read without zooming in?
- Do you refer at least once to each figure in your text?
- Is it clear what each component in your model represents?
- Did you use citations where necessary?
- Are your citations formatted correctly?