2019•2020

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterthesis

Comparison of PI/PID and model predictive control applied to a Simscape Multibody model of a DC-powered conveyor belt

PROMOTOR :

Prof. dr. ir. Johan BAETEN

PROMOTOR :

Prof. dr. ir. Juhani HENTTONEN

Ruben Berden, Tristan Olaerts

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie, afstudeerrichting automatisering

Gezamenlijke opleiding UHasselt en KU Leuven

UHASSELT KU LEUVEN

UHASSELT KU LEUVEN

2019•2020

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterthesis

Comparison of PI/PID and model predictive control applied to a Simscape Multibody model of a DC-powered conveyor belt

PROMOTOR :

Prof. dr. ir. Johan BAETEN

PROMOTOR :

Prof. dr. ir. Juhani HENTTONEN

Ruben Berden, Tristan Olaerts

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie, afstudeerrichting automatisering

UHASSELT KU LEUVEN

Deze masterproef werd geschreven tijdens de COVID-19 crisis in 2020. Deze wereldwijde gezondheidscrisis heeft mogelijk een impact gehad op de opdracht, de onderzoekshandelingen en de onderzoeksresultaten.

FOREWORD

This thesis was written as the concluding chapter of the study Master of Science in Energy Engineering Technology with a focus on automation technology. This education is part of the joint programme of UHasselt and KU Leuven. The master's thesis was carried out at HAMK University during an Erasmus exchange project. Unfortunately, the exchange was stopped prematurely due to the outbreak of the COVID-19 virus in Europe but the project was completed remotely.

The choice of the subject of the master's thesis stems from the demand of the industry for better knowledge of automation engineers on the application of control mechanisms. This thesis would not have been realized without the help, support, cooperation and time of many people. We would like to thank them in this way.

First of all, we would like to thank our promoters and supervisors, prof. dr. ir. Juhani Henttonen and prof. dr. ir. Johan Baeten. They spent their valuable time answering our questions and keeping us in the right direction. They gave us valuable tips and information to improve and complete the project. Furthermore, we would like to thank Ms. Greet Raymaekers and dr. Els Wieërs for giving us the opportunity to study abroad and to give us the time and effort to make this experience possible. Special thanks to Ms. Leena Koivisto for shaping our Erasmus experience in Finland and helping us with our stay.

Finally, we would like to thank our parents for their support and trust. They have enabled us to get a higher education. Not to forget the people who supported us during our studies. Hereby we would like to thank brothers and sisters, partners, family and friends. They are people of support and rest for us. Thanks to them we came to the end of our study, the writing of the thesis.

Ruben Berden
Tristan Olaerts

Diepenbeek, 2 juni 2020

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This research is commissioned by the Electrical and Automation Engineering Department of HAMK Häme University of Applied Sciences in Finland. Designing models and controllers to predict the behaviour of systems is becoming increasingly important in the industry. Choosing the right controller is an important factor as well. The aim of this thesis is to investigate the toolchain from CAD model to a Simulink model and to compare some of the most frequently used controllers and make a founded choice for the conveyor belt application.

The project is applied to a simple DC-powered conveyor with PWM control. This installation is mechanically modelled in Onshape. This model is then exported to the Simscape Multibody environment. Next, a friction model is applied. Subsequently, two types of controllers are designed for the model of the installation: PI/PID and model predictive control. These controllers and their operation are compared with each other.

The conversion from Onshape to Simscape Multibody is extremely straightforward. However, the choice of parameters to match the model to reality was impossible due to the coronavirus. Simulations have shown that for the conveyor application a gain scheduled PID controller performs better than a single MPC controller at speed control because of the multiple linearizations at multiple operating points. In position control, PID and the MPC control are equivalent.

ABSTRACT IN DUTCH

Dit onderzoek is uitgevoerd in opdracht van de afdeling Elektrotechniek en Automatisering van HAMK Häme University of Applied Sciences in Finland. Het ontwerpen van modellen en regelaars om het gedrag van systemen te voorspellen wordt steeds belangrijker in de industrie. Ook de keuze van de juiste regelaar is een belangrijke factor. Het doel van deze masterscriptie is het onderzoeken van de *toolchain* van CAD-model naar een Simulink model en het vergelijken van veelvoorkomende controlestrategieën en het maken van een gefundeerde keuze voor de transportbandtoepassing.

Het project is toegepast op een eenvoudige DC-aangedreven transportband met PWM-sturing. Deze installatie is mechanisch gemodelleerd in Onshape. Vervolgens is dit model geëxporteerd naar de Simscape Multibody omgeving. Daarna is een wrijvingsmodel ontworpen en toegepast. Dan zijn voor het complete model van de installatie twee regelaars ontworpen: PI/PID en *model predictive control*. Tenslotte zijn deze regelaars en hun werking geanalyseerd en met elkaar vergeleken.

De overzetting van Onshape naar Simscape Multibody is zeer eenvoudig. Echter werd de parameterkeuze om het model overeen te laten komen met de werkelijkheid onmogelijk gemaakt door het coronavirus. Simulaties toonden aan dat voor de transportbandtoepassing een *gain scheduled* PID-regelaar beter presteert dan een enkele MPC-regelaar bij snelheidsregeling vanwege de meervoudige linearisaties bij meerdere werkingspunten. Bij positieregeling zijn PID en de MPC-regeling gelijkwaardig.

1 INTRODUCTION

1.1 SETTING

This master's thesis is carried out by two students of the joint programme Master of Science in Energy Engineering Technology of UHasselt and KU Leuven. The project is conducted at the Hämeen Ammattikorkeakoulu (HAMK) during an Erasmus exchange and is executed in the Electrical and Automation Engineering department located in Valkeakoski, Finland. In the university's laboratory, there is a simple conveyor belt, shown in figure 1, which is modelled and controlled. The conveyor belt is straight and horizontal and is equipped with a 24V DC motor that can be controlled in speed using PWM control. The CAD model is implemented in Matlab Simscape Multibody. In this environment, different control algorithms are compared with each other: PI/PID control and model predictive control. Nowadays, model predictive control is used more and more in practice to facilitate automation. There are several reasons for the use of model predictive control. All kinds of constraints (actuator limits, output limits) can be taken into account during control calculation. If a nonlinear process model is available, nonlinear control can be applied and complex systems can be controlled because first a simpler model of the system is created which is used for control.

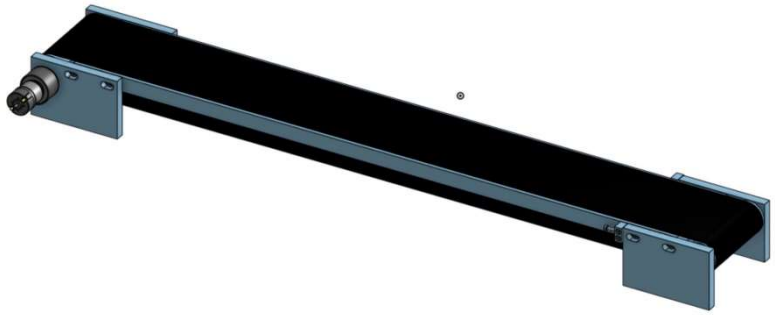


Figure 1: 3D representation of the conveyor belt

1.2 PROBLEM ANALYSIS / RESEARCH QUESTION

This study is set out to investigate three main aspects of the modelling and control of a conveyor belt. The first objective is to investigate how well the toolchain from CAD to Simulink Multibody to real-time control works. The quality of the toolchain is evaluated in terms of simplicity, amount of errors and amount of manual modifications. The second part is to examine how easy or difficult it is to identify the parameters of a Simulink model. The third objective is to determine how much benefit model predictive control can provide over PI/PID and whether the available simplified implementations of model predictive control are sufficient.

1.3 OBJECTIVES

The main objective of the thesis is to be able to carry out all the proposed steps and form a well-founded conclusion within the given timeframe. This objective is achieved if, first of all, a model can be made that corresponds to reality within small margins. The model must take into account the actual friction. Furthermore, at least three different working regulators have to be made. Based on simulation values and performance results, a well-founded choice must be made between the different controllers.

1.4 METHOD

The 3D model of the conveyor belt is drawn using the free online tool Onshape. This tool is used because Onshape is compatible with Simulink Multibody which makes it possible to implement an Onshape drawing with a few simple commands. Once the 3D model is implemented in Simulink Multibody, a friction model is added. First, some friction models are researched and the best one is chosen. This model is then applied to the Simulink model to become a realistic conveyor model. Small adjustments are made to assure that the model is plausible. Next, control algorithms are designed for the conveyor belt. The two different used algorithms are PI/PID and model predictive control. The behaviour of these different controllers is compared to each other in the Simulink Multibody environment. Once the control algorithms are optimized, simulations are carried out to compare the different controllers. Lastly, the results are compared in order to draw a conclusion.

1.5 THESIS STRUCTURE

This thesis consists of several parts which are divided in different chapters. Following the introduction, the next chapter contains the literature study which provides insight into existing techniques and applications. Note that neural network control, a control algorithm that occurs regularly, is also been researched and discussed. However, the focus of this thesis is on MPC and PI/PID and, therefore, neural network control is not further discussed.

Next, the step-by-step construction of the complete model is discussed in chapters three to seven. This includes the graphical model, the conversion from graphical model to Simulink Multibody, the motor model, the friction model and finally an overview of the total model.

In chapter eight, the different controllers from the literature study are applied to the conveyor belt. Next, their behaviour is compared in chapter nine and, finally, the most important conclusions are described in chapter ten.

2 LITERATURE STUDY

One of the goals of the master's thesis is to compare different control mechanisms applied to a conveyer belt. First, in this literature study, different friction models are researched to obtain a realistically possible model. Afterwards, different controller techniques are investigated to be able to apply these to the model.

2.1 FRICTION MODEL

The conveyor belt model should be a good representation of the reality. Therefore, friction should be taken into account. Numerous friction models have been developed for motion control, each with its own behaviour. In this section, several basic friction models are described. This knowledge is then used to make a founded choice of the friction model. Note that due to the COVID-19 quarantine, the social situation made testing of the final model impossible. As a result, only the simple models that should give a plausible result are discussed.

2.1.1 STATIC FRICTION

Static friction occurs when the objects do not move relative to each other. The coefficient of static friction is usually higher than the coefficient of kinetic friction. Static friction is considered to be the result of surface roughness properties over several length scales on solid surfaces. Static frictional force must be overcome by an applied force before an object can move. Mathematically, this translates into the following expression:

$$F = \begin{cases} F_C \,.\, Sgn(v) & if\ v \neq 0 \\ F_{app} & if\ v = 0\ and\ F_{app} < F_C \end{cases} \quad (1)$$

where F is friction force, v is sliding speed, F_{app} is applied force and F_C is the Coulomb friction force [1], [2], [3].

2.1.2 COULOMB FRICTION

The Coulomb friction force is defined as:

$$F_C = \mu \,.\, F_N \quad (2)$$

where μ is the Coulomb friction coefficient or the dynamic friction coefficient and F_N is the normal force between the contact surfaces. The Coulomb friction model is shown in figure 2. When $F_{app} < F_C$, there is no sliding ($v = 0$). In this case, the Coulomb force can take any value from zero up to maximum $\mu \,.\, F_N$. When $v \neq 0$, the Coulomb friction force takes the value F_C or $-F_C$ depending on the sliding direction [1]-[3].

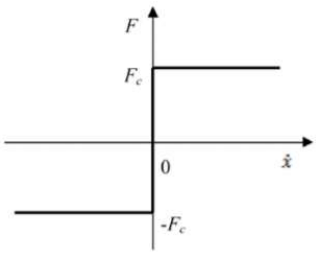


Figure 2: Coulomb friction model [1, p. 17]

2.1.3 VISCOUS FRICTION

The viscous friction force is defined as:

$$F = k_v . v \tag{3}$$

where k_v is the viscous coefficient and v is sliding speed. As shown in figure 3, the viscous friction force is a linear function of the sliding velocity. The viscous friction model is limited because of the inaccurate representation in regions without lubricant [1]-[3].

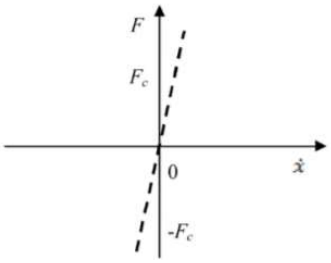


Figure 3: Viscous friction force [1, p. 17]

2.1.4 STRIBECK FRICTION

The Stribeck friction force is described as:

$$F = \left(F_C + (F_S - F_C).\, e^{-\left(\left|\frac{v}{v_s}\right|\right)^i} \right).\, Sgn(v) + k_v.\, v \tag{4}$$

where F is friction force, v sliding velocity, F_C the Coulomb friction force, F_S the static friction force, v_S the Stribeck velocity, k_v the viscous friction coefficient and i an exponent. This model has for a certain velocity regime a decreasing friction force with increasing velocity. This is called the Stribeck effect and is shown in figure 4. This representation proves accurate for real friction. When $i = 1$, the Tustin's model is described. It is one of the best models to describe friction at near zero speeds [1]-[3].

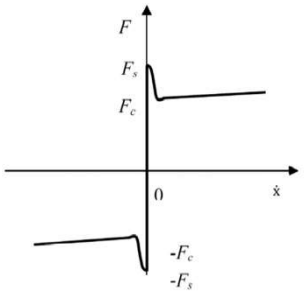


Figure 4: Stribeck friction model [1, p. 18]

2.2 PID CONTROL

PI/PID controllers are widely used in the process industries and commercial controller hardware. They are also used in a non-model based friction compensation, which can eliminate stick-slip in high-stiffness systems. PID controllers are widely used in servo drive systems based on cascade principle as current, speed or position controllers.

2.2.1 PI controller

The PI controller is a combination of the P- and I controller. First, the P- and I controller are explained.

- P controller

The operation of a P controller is best explained with the help of figure 5. This figure shows a control loop with a P controller. In this figure, x is the setpoint and must be set to the value that should be reached by the controller. A feedback is used to compare the setpoint with the output value. From this comparison comes the control deviation (e). In other words, the difference between the set value and the process value. The error signal e (or control deviation) is simply amplified (or reduced) to generate the control signal for the system.

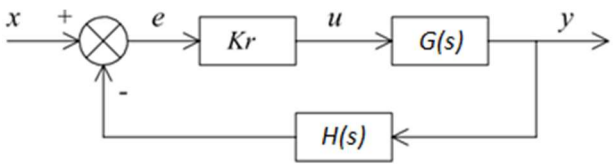


Figure 5: Control loop with a P controller [4, p. 4.1]

The transfer function of a P controller is the following:

$$G_P = K_r \tag{5}$$

A disadvantage of the P controller is that it has a position error. So, the amplification factor should not be chosen too small or too large as this will result in too much deviation or system oscillation, respectively. When choosing the value for the gain factor K_r, a compromise has to be made between speed and accuracy.

- I controller

As stated above, the P controller has a position error. The I regulator however is able to make the position error zero. A P controller only takes the current value of the deviation into account and bases its control value on this whereas the I controller will also take the deviation from the past into account (Integrator). As long as the control difference is not zero, the controller will increase (or decrease) its control signal to obtain an adjustment. If the error becomes zero, the control value will not change but will also not fall to zero. This allows the controller to fully compensate for the position error in a closed circuit. An I controller is shown in figure 6.

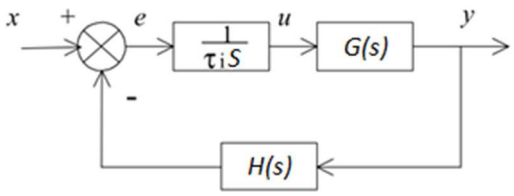


Figure 6: Control loop with a I controller [4, p. 4.3]

The transfer function of the I controller is the following:

$$G_I = \frac{1}{\tau_i s} \tag{6}$$

The time constant τ_i in this equation indicates how fast the integration takes place. A small value gives a fast integration but can have the disadvantage that the system becomes unstable. A high value gives a slow integration and has the disadvantage that the system becomes slow.

- PI controller

In the previous two bullets the P and I controllers were explained. This makes it easier to explain the working principle of the PI controller. First, the transfer function of the proportional integrator (PI) controller is described.

$$G_{PI} = K_r \left(1 + \frac{1}{\tau_i s}\right) \tag{7}$$

In the transfer function, one can clearly distinct the transfer function of the P- and I action. So, the PI controller is a combination of these two actions. The PI controller tries to combine advantages of both methods at the same time minimizing the disadvantages. For example, for a fast response and a small error the P controller needs a high value of gain (K_r). This will eventually lead to oscillations in the control circuit and also random measurement noise is amplified. On the other hand, the I controller is nearly immune to high frequency random noise and can remove the constant error completely. The problem with the I controller is that a fast response is not possible because for small values of the integration time, the control system will be oscillatory and at some point unstable. So, a PI controller tries to combine the best parts of each component. In figure 7 the step response of a random PI controller is shown [4].

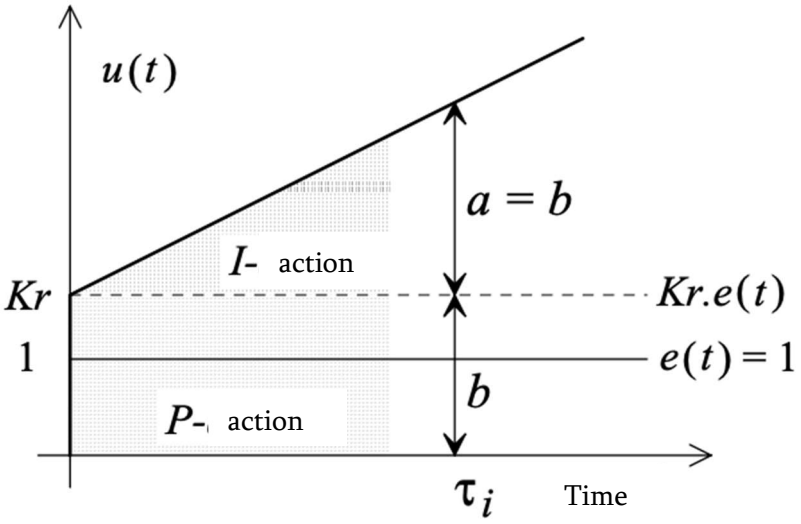


Figure 7: Step response PI controller [4, p. 4.5]

2.2.2 PID controller

- Differential action

In addition to the P and I action, there is also a third rule action, which is called the D action. As the name indicates, this action will become active the moment there is a change in the control error. In other words, the D action responds to the speed at which the control error changes. A slow change results in a small control contribution, a fast change in a large control contribution. So, the D action can react very violently and in theory can give an infinitely large and very narrow impulse at a step change. This behaviour is visible in figure 8.

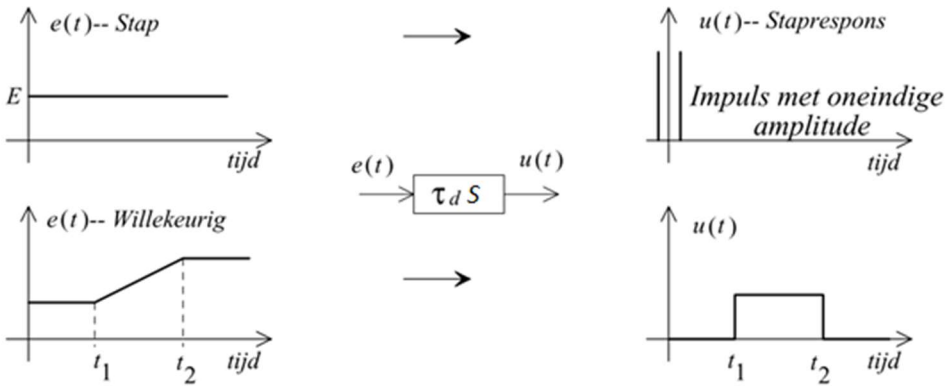


Figure 8: A random D action [4, p. 4.13]

The transfer function of the D action is the following:

$$G_D = \tau_d s \tag{8}$$

The differentiation time τ_d in this equation tells something about the differentiation. With a small value, the influence of the D action will be very low or even nonexistent. With a value that is too high, the controller will become restless.

In practice, a pure differentiator rarely occurs on its own and, therefore, usually forms part of a larger whole.

The D action has a stabilizing effect on the control loop.

- PID controller

The PID controller combines the advantages and disadvantages of the P-, I- and D action.

The most common types of PID controllers are the parallel and the serial PID controller. They have about the same effect on the control loop or on the system. Only the influence of the time constants is slightly different for the two regulators.

- Parallel PID controller

The transfer function of the parallel PID is the following:

$$G_{PIDp} = Kr\left(1 + \tau_{dp}s + \frac{1}{\tau_{ip}s}\right) \tag{9}$$

- Serial PID controller

The transfer function of the serial PID is the following:

$$G_{PIDs} = Kr(1 + \tau_{ds}s).\left(1 + \frac{1}{\tau_{is}s}\right) \tag{10}$$

Often, for the sake of simplicity, the parallel PID controller will be used if the control is based on the step response (or a time response). However, if the frequency plot is needed, preference will be given to the serial PID controller which has a concatenation of a PI and a PD controller. Figure 9 shows the step response of a random parallel PID controller [4].

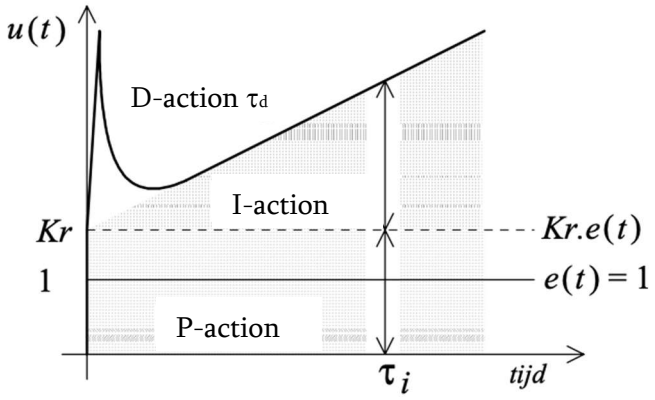


Figure 9: Step response parallel PID controller [4, p. 4.16]

2.3 MODEL PREDICTIVE CONTROL

Model Predictive Control is a control strategy that uses the dynamic system model. The basic idea behind this strategy is clarified by figure 10. Reference trajectory generation is part of the control algorithm. It utilizes knowledge of how the system should behave in the future. At the given moment *t*, the current state of the setup is sampled. From this state, the ideal path is calculated by minimizing the cost function. This path is the output prediction trajectory *z(t/k)*. It describes how the reference path should be achieved by the output signal. When a set of future inputs is generated, only the first one is executed and the cycle is repeated.

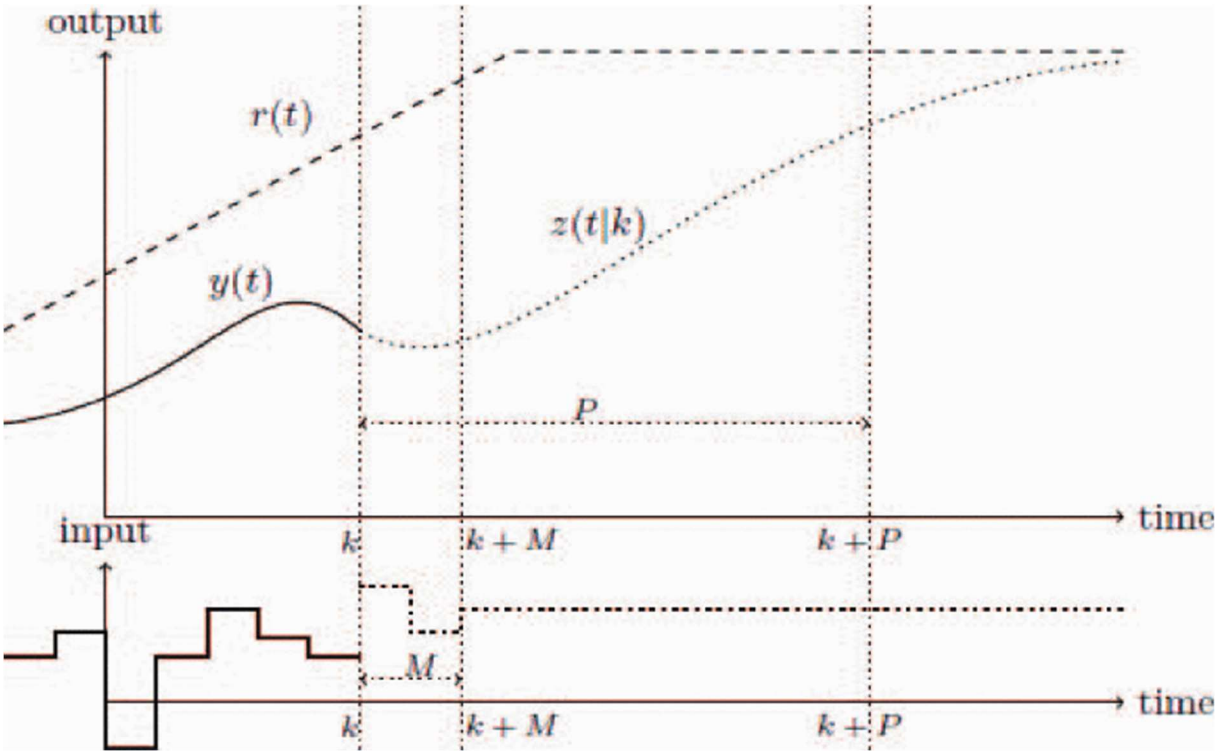


Figure 10: Basic idea behind model predictive control [6, p. 3]

The output prediction trajectory is defined over a specified number of future samples. This is known as the prediction horizon P. The inputs of the system are responsible for driving the setup along the prediction trajectory. These are assumed to change over a specified number of samples. This is called the control horizon M. After the control horizon, the inputs are assumed to stay constant. Using the dynamic system model, the output behaviour of the system is determined within the prediction horizon P. Control signal values are calculated so that a cost function is minimized. This cost function (11) takes the desired behaviour of the system into account [5]–[7].

$$J = \sum_{j=1}^{P} [\hat{y}(k+j) - r(k+j)]^2 + \rho \sum_{j=1}^{M} [\Delta u(k+j-1)]^2 \quad (11)$$

2.4 NEURAL NETWORK CONTROL

In recent years, the interest in neural network has expanded. Artificial neural network was developed based on the neuron structure. A basic mathematical model of a neural network is built in three layers. These are the input layer, the hidden layer and the output layer. These layers have a simple parallel computation layer but have an interesting learning ability and computational power to predict dynamic patterns. Neural networks can approach any unknown continuous nonlinear function by overlapping the outputs of each neuron. So, by choosing the correct number of neurons the approximation error can be made as small as possible.

2.4.1 Types of neural networks

Different types of neural networks structures, which are commonly used in the control engineering, are introduced in this section.

2.4.2 Radial Based Function Neural Network (RBFNN)

The basic structure of a RBFNN network is given in figure 11. The three different layers that where mentioned earlier are clearly visible. These are the input layer, the hidden layer and the output layer. Z is the neural network input vector which is applied on the input layer. The hidden layer transforms the data from the input space to a space with a higher dimension. This is the hidden space. This type of neural network can be used to approach continuous vector functions [8].

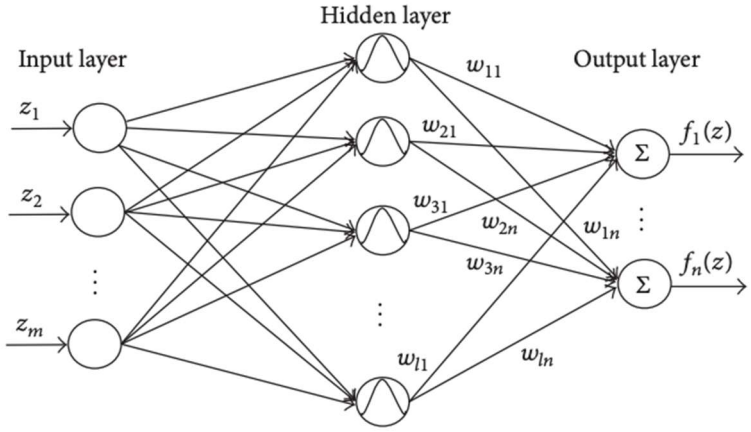


Figure 11: Structure of RBFNN [8, p. 3]

Following an example from [8],

$$F(Z): F(Z)^* = W^T S(Z) \tag{12}$$

F(Z)* is the estimation of $F(Z)$ and Z is NN inputs vector. The optimal weight between the hidden layer and the output layer is estimated as follows:

$$W^* = [W_1^*, W_2^*, \dots, W_n^*] \in R^{n\,x\,l} \tag{13}$$

and

$$S(Z) = [s_1(Z), s_2(Z), \dots, S_n(Z)]^T \tag{14}$$

the regressor and l represent the number of neural network nodes. Generally, the regressor could be chosen as a Gaussian radical basis function as follows:

$$s_i(\|Z - u_i\|) = \exp\left[\frac{-(Z-u_i)^T(Z-u_i)}{\sigma_i^2}\right] \tag{15}$$

where u_i (i = 1,…, l) are distinct points in state space and σ_i is the width of Gaussian membership function. It has been well recognized that, using the powerful approximate ability of the RBFNN, any continuous nonlinear function can be approximated over a compact set as

$$F(Z) = W^{*T} S(Z) + \varepsilon \tag{16}$$

where W^* is the optimal weight vector and ε is the approximate error.

2.4.3 Cerebellar Model Articulation Controller (CMAC) neural network

The Cerebellar Model Articulation Controller was first proposed by James Albus as a learning mechanism that was based on a cerebellum neurophysiological model and that imitates the structure and function of the cerebellum. CMAC neural networks have been used extensively for modeling and control of robot systems because of its rapid learning speed, simple structure, insensitivity of data sequence an easy implementation. Figure 12 displays the architecture of a CMAC neural network [8].

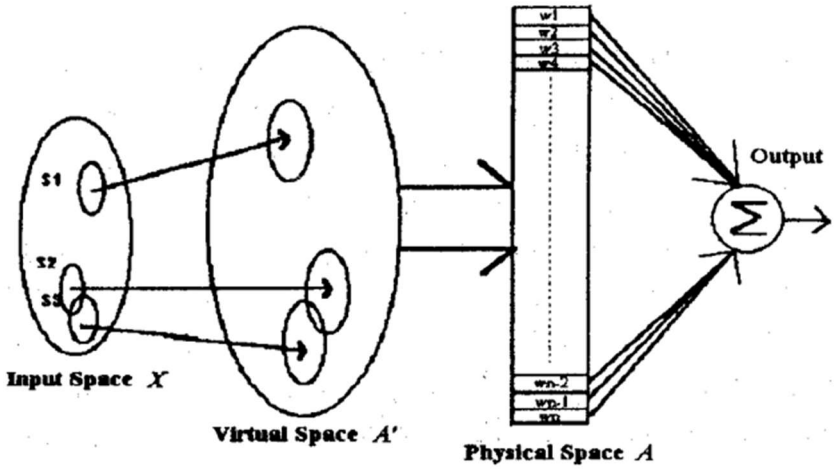


Figure 12: The architecture of a CMAC neural network [9, p. 1417]

A basic CMAC neural network is usually considered as a two-layer association neural network. It consists out of two basic mapping functions:

- X → A'
- A' → A

These two basic mapping functions are visible in figure 12. The first function represents each point in the input space X onto a vector into the virtual space A'. The virtual space will grow to large in practice in most cases. Due to this reason, a hash-code technique must be used to compress the virtual space A' into an physical phase A. If the above-mentioned procedure is used, each point on the input vector is mapped onto a vector of adjustable weights in the physical space A.

$$W = [w_1, w_2, \dots, w_n] \tag{17}$$

The output of the CMAC neural network is equal to the sum of all the weights [9].

$$Y^j = \sum_{i=1}^{c} w_i^j \tag{18}$$

3 MECHANICAL DESIGN

This section describes the mechanical model of the conveyor belt. The CAD design is made using the free online tool Onshape. It is used because of its compatibility with Matlab Simscape Multibody. The isometric view of the conveyor belt assembly is shown in figure 13.

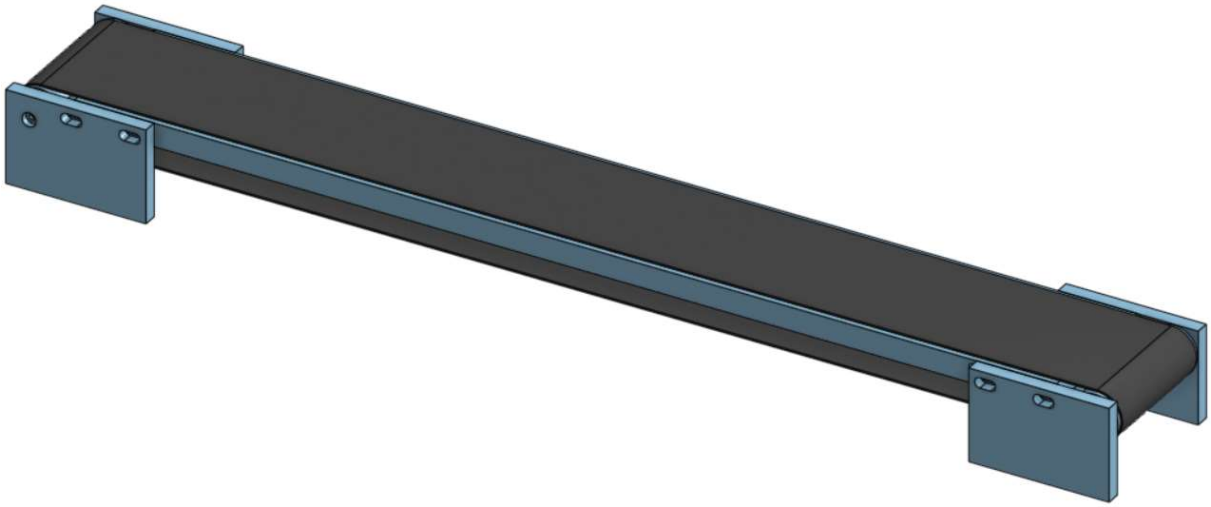


Figure 13: Isometric view of the conveyor belt assembly in Onshape

3.1 GRAPHICAL MODEL

The graphical model consists of eight parts which are illustrated in figure 14. Note that a mounting plate has been hidden to make the roll visible. There are four mounting plates (4) of which one has a cut-out to mount the DC motor. Next, there is a support plate (1) which ensures that the belt (3) itself does not deflect due to gravity. The belt is tensioned on two rolls (2). The motor exerts its torque on one of these two rolls. The drawings of these parts can be found in attachment A. Note that the 2D drawings contain a clamping plate. This plate is used to tension the belt, but it is left out in the model for simplicity.

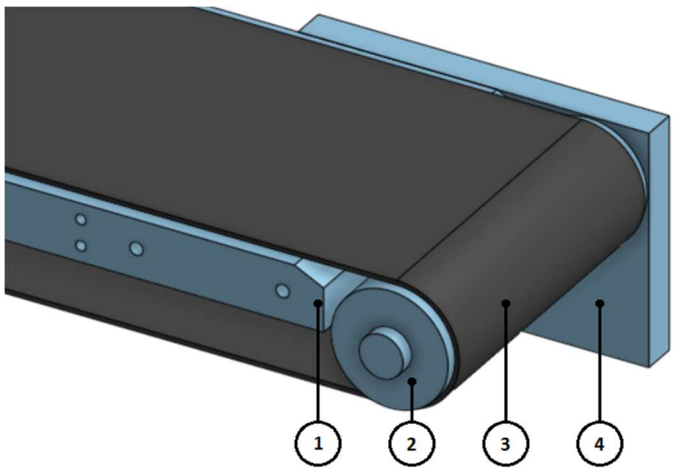


Figure 14: Isometric view of the conveyor belt parts in Onshape

3.2 DC MOTOR

The conveyor belt is controlled by a DC motor with PWM-modulation. This motor is from the RH158 series of the brand Micro Motors. The technical data can be found in attachment B. The most important specifications are summarized in table 1. The motor is shown in figure 15. In the conveyor belt application, the motor housing is attached to mounting plate 1. The shaft of the motor goes through this plate into the roll. Here, it exercises the torque that will cause the rotation [10].

Table 1: Technical data of the RH158.24.75 DC motor [10]

Nominal voltage	Nominal torque	Speed no load	Speed at nominal torque	Current no load	Current at nominal torque	Input power
V	Ncm	rpm	rpm	mA	mA	W
24	50	81	55	<70	340	8,2

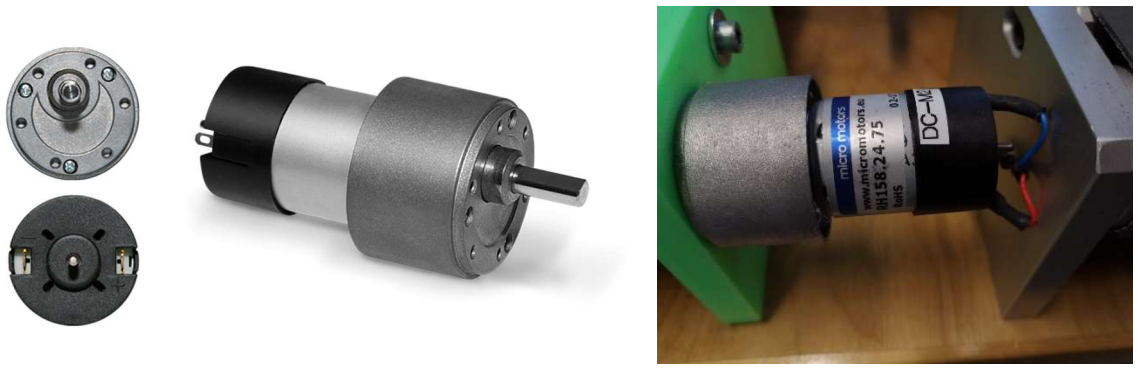


Figure 15: Picture of the RH158.24.75 DC motor [10]

4 CONVERSION TO SIMULINK MULTIBODY

As stated in chapter three, the mechanical model is made in Onshape. This 3D drawing software works analogous to common software packages of Autodesk and SolidWorks. Onshape models are similar in structure to other multibody models. Parts connect with each other by means of mates such as Ball, Slider and Revolute to form assemblies. When this model is built correctly with the right restrictions on the degrees of freedom, it can be exported to the Simscape Multibody environment.

4.1 WORKFLOW

Importing the Onshape model into the Simscape Multibody environment is possible using the 'smexportonshape' and 'smimport' functions. The smexportonshape function converts the Onshape model into an intermediate representation comprising an XML file and a set of STEP files. These generated files are needed to import a model from Onshape. The XML file identifies the bodies of the model and defines their kinematic relationships. The STEP files provide the 3D geometries of the parts. The smimport function converts the XML file into the Simscape Multibody model and a supporting data file. A schematic workflow of this process is shown in figure 16. Figure 17 shows the Matlab code when the functions are implemented [11].

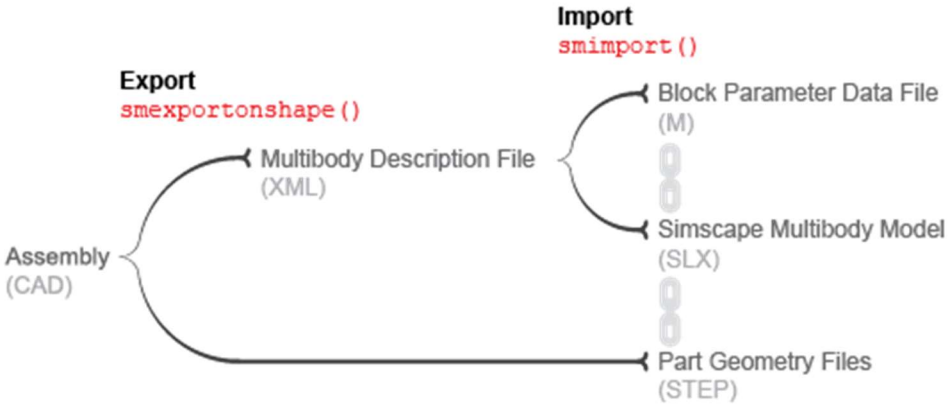


Figure 16: Onshape CAD import workflow [11]

```
Editor - C:\Users\trist\OneDrive\Documenten\MATLAB\Onshape.m
Onshape.m
1 -    url = 'https://cad.onshape.com/documents/73fa61fb5e4525bd6e50ecfe'
2 -    xmlFile = smexportonshape(url)
3 -    smimport(xmlFile)
```

Figure 17: Matlab code to import Onshape CAD model

4.2 SIMSCAPE MULTIBODY MODEL

The execution of the Matlab code provides the Simscape Multibody model shown in figure 18. Immediately, different parts can be distinguished. First, there is the *world frame block* together with the *mechanism configuration block* and the *solver configuration block*. These blocks are needed as a spatial reference, to set gravity and to set the solver parameters for the simulation. Next, there are the different parts from the CAD model. These parts are connected with the world frame through the *transform block.* This block is connected to mounting plate 2 because this part was set as fixed in Onshape. Finally, there is the *revolute joint block* between the two rolls and the mounting plates. This indicates one rotational degree of freedom between the frames of each part.

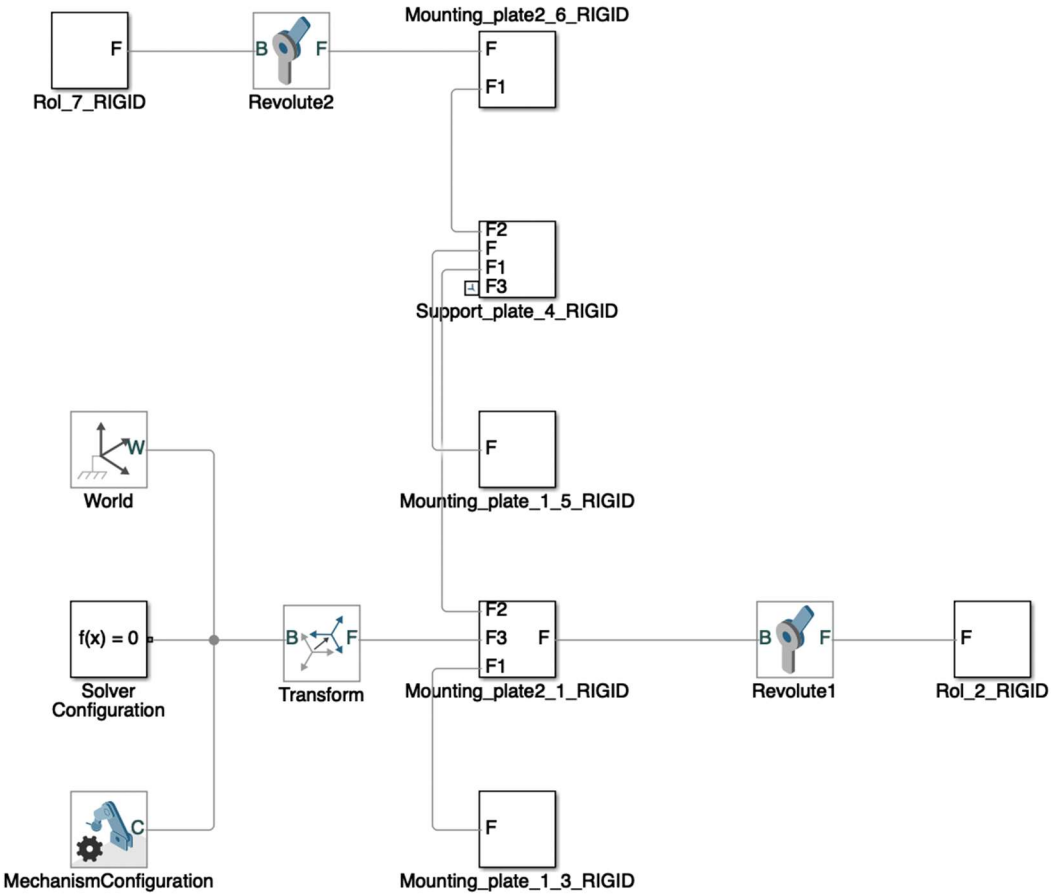


Figure 18: Generated Simscape Multibody model

It is very simple to convert an Onshape model to a Simulink Multibody model as illustrated above. If the CAD model is assembled properly without any geometrical errors and with correct degrees of freedom, the Simulink model will be generated correctly. This challenge of the CAD model would be just as important if the model had to be built directly in Simulink. However, by building it in a familiar CAD environment, the design process is accelerated.

5 MOTOR MODEL

In this section, the structure of the motor model in Simulink Multibody is discussed. In the previous part, the Simulink model was generated from Onshape. However, this model has no actuator yet. The motor of the installation is a 24V DC motor whose voltage is controlled by PWM modulation. The properties of the motor can be found in section 3.2. The motor model is shown in figure 19. Two parts can be distinguished: the electrical portion in blue and the mechanical portion in green.

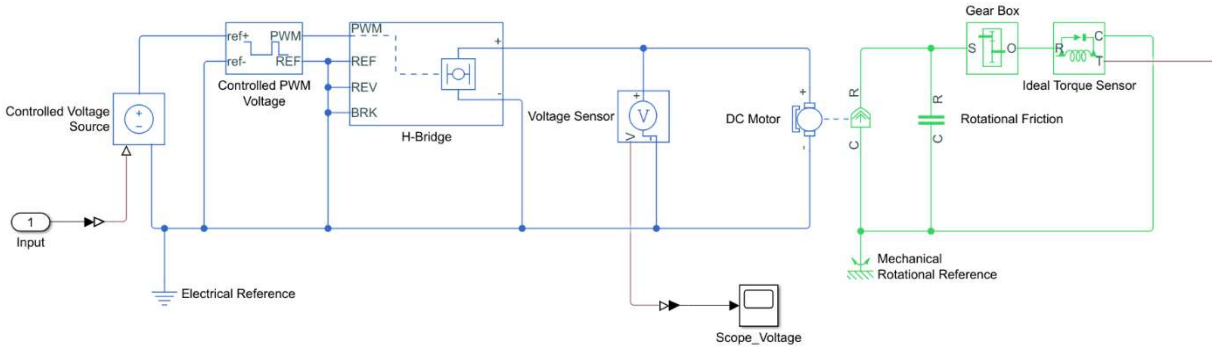


Figure 19: DC motor model with PWM modulation

5.1 ELECTRICAL MODEL

The electrical portion consists of several blocks which are further discussed in this section. First, the voltage is set by the *Controlled Voltage Source block*. This block represents an ideal voltage source where the output voltage is the numerical value presented at the physical signal port. This physical signal port is connected via a *Simulink-PS Converter* to the input of the subsystem. The manipulated variable of the controller will be connected to this input [12].

Next, the Pulse-Width Modulated (PWM) voltage is created across the PWM and REF ports in the *Controlled PWM Voltage block*. The output voltage is 24V when the pulse is high, and is 0V when the pulse is low. The 24V is set by the Output voltage amplitude parameter. Duty cycle is set by the input value. This indicates a 50% duty cycle when the input value is 12V.

This PWM signal is applied to the *H-Bridge block.* The simulation is in Averaged mode for faster simulation time. In this mode, the ratio of the on-time to the PWM period is defined by the PWM port voltage divided by the PWM signal amplitude parameter. Using this ratio and assumptions about the load, the block applies an average voltage to the load that achieves the correct average load current.

The output signal of the *H-Bridge block* is fed to the *DC motor block*. This block represents the electrical and torque characteristics of a DC motor. It assumes no electromagnetic energy is lost, so all electrical energy is converted into mechanical energy. The RH158.24.75 DC motor characteristics are implemented in the parameters of this block. The output signal of the H-Bridge is monitored by the scope by means of the *Voltage Sensor block*. It represents an ideal voltage sensor. All blocks have been referenced to the ground by the *Electrical Reference block* [13].

5.2 MECHANICAL MODEL

The mechanical portion, shown in figure 20, consists of several blocks which are further discussed in this section. The mechanical side of the *DC Motor block* represents the mechanical connections of the motor. The *Mechanical Rotational Reference block* is connected to the stationary part of the motor: the housing. Next, the friction of the bearing in the motor is modelled using the *Rotational Friction block*. The *Gear Box block* is implemented to characterize only one parameter, the Gear ratio. This ratio is 76,84 for the RH158.24.75 DC motor, as found in the technical sheet in attachment B. The final block in the mechanical portion of the motor is the *Ideal Torque Sensor block.* This block is used to measure the applied torque and transfer it to the revolute joint of the model. This connection is discussed in the next section.

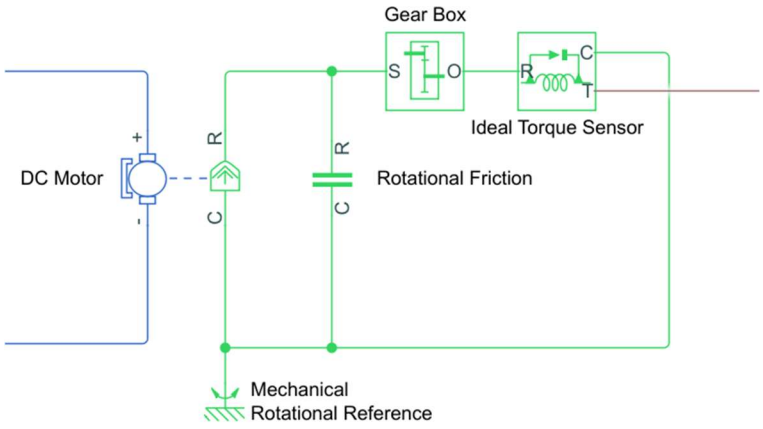


Figure 20: Mechanical portion of DC motor model with PWM modulation

5.3 MOTOR CONNECTION TO REVOLUTE JOINT

The mechanical portion of the motor model is discussed in section 5.2. The *Ideal Torque Sensor* provides the torque signal that is used to connect the motor to the roll. The signal is applied to the *Revolute Joint 2 block,* as shown in figure 21.

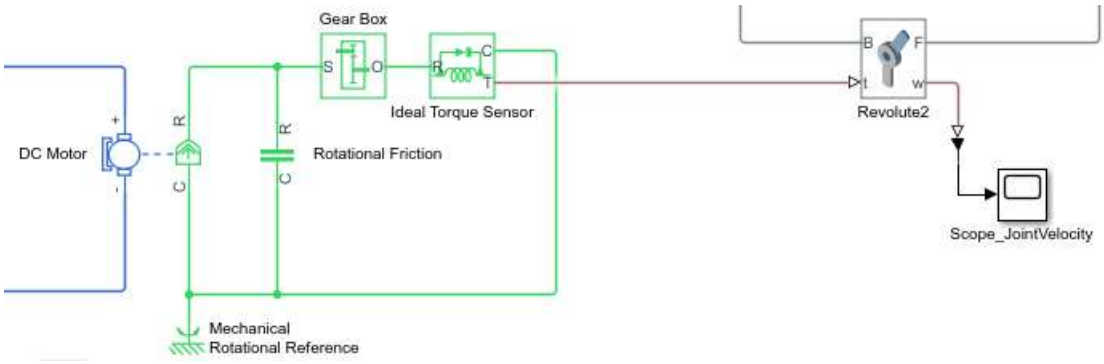


Figure 21: Connection DC motor to Revolute Joint

The input port of the revolute joint is not present by default. This requires the properties to be adjusted as shown in figure 22. In the Actuation tab, the torque should be Provided by Input. This creates the extra port in the model. In the Sensing tab, an angular velocity output signal is created by checking the box. When a scope is added, this provides a visual tool to see if the conveyor belt is actually working.

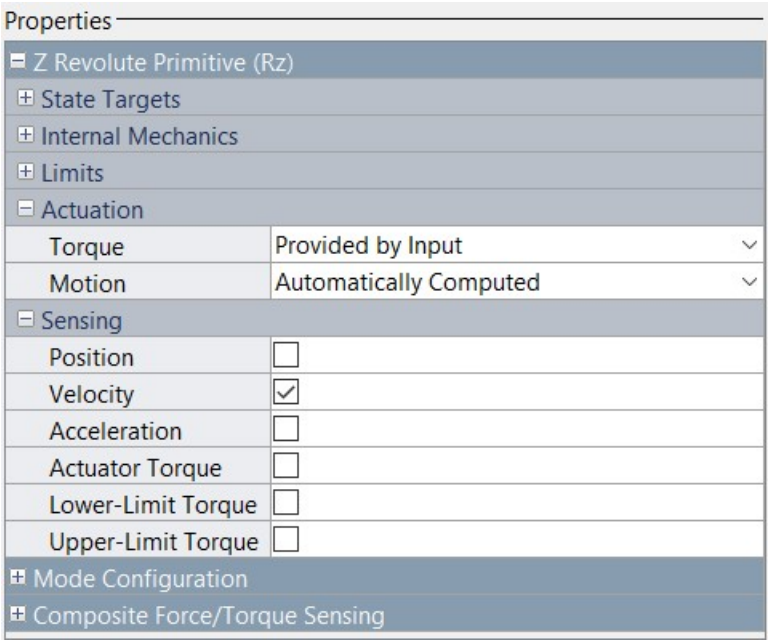


Figure 22: Properties of Revolute Joint 2

6 FRICTION MODEL

This chapter describes how friction is implemented in the model. As mentioned in the literature study, there are different methods to describe friction. For the conveyor belt application, the Stribeck friction model has been chosen. This is because it is an advanced model that is quick to implement in Simulink Multibody. It also corresponds well with reality. Note that due to the coronavirus outbreak in Europe, testing the friction parameters was impossible. Therefore, the parameters used in the model are only an estimate.

The friction is applied to the parts that move relative to each other. For the conveyor belt, these are the revolute joint connections and the belt that slides over the support plate. Auxiliary blocks from the Matlab Multiphysics library have been used to implement the friction blocks in the model. This is the *Rotational Simscape Multibody block* as shown in figure 23. This block is the interface to add the mechanical blocks to the model.

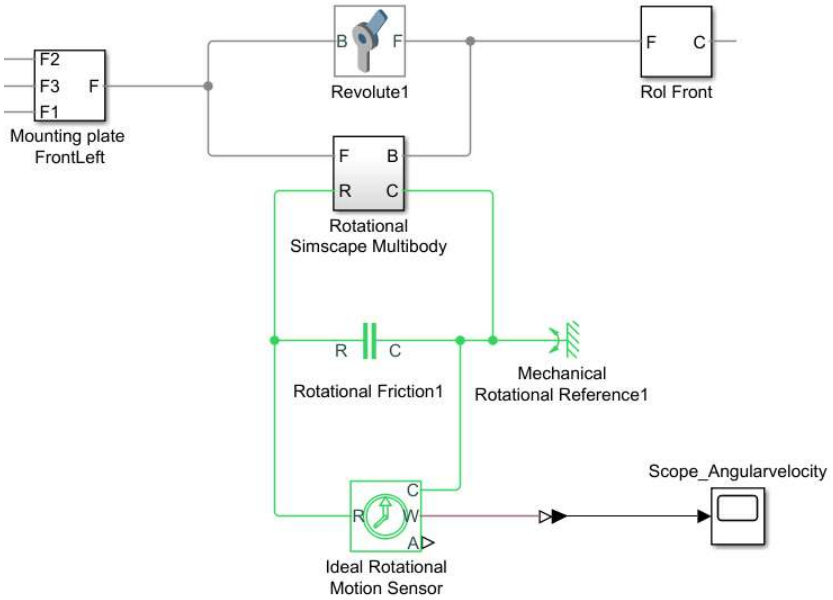


Figure 23: Implementation of friction in the conveyor belt model

First, the *Rotational Friction block* represents friction in the contact between rotating bodies. The friction force is simulated as a function of relative velocity and assumed to be the sum of Stribeck, Coulomb, and viscous components. Next, the *Mechanical Rotational Reference block* signifies the mounting plate as a reference for the rotation. In this way, the correct direction of movement is determined and the speed can be measured. This is done by the *Ideal Rotational Motion Sensor block*. This device converts an across variable measured between two mechanical rotational nodes into a control signal proportional to angular velocity. This construction is analogous to the friction at the other revolute joint and the friction between the support plate and the belt. However, for the latter the friction is not rotational but translational. So the *Translational Friction block* and the *Mechanical Translational Reference block* are used respectively [14], [15].

7 FINAL MODEL

The model is almost complete: the parts have already been imported from Onshape, the motor model has been built up and connected and the friction has been added. Now, only the speed and position of the conveyor belt needs to be measured. This objective is realized in this chapter.

7.1 MEASURED VARIABLES

Section 5.1 describes how the input voltage of the motor can be set by the controller. The controller also needs feedback from the system. This feedback is called the measured variable of the plant. In case of the conveyor belt, the measured variable is the belt speed. Note that the position can also be used as a measured variable by integrating the speed signal.

The angular velocity is measured by the sensor block shown in figure 24. This velocity is the rotation speed ω in rad/s. To convert this to translational speed, the following formula is used:

$$\vec{v} = \vec{\omega}\, x\, \vec{r} \tag{19}$$

where r is the radius of the roll equal to 2 cm or 0,02 m. This translates in Simulink to a gain block with k = 0.02. The signal now has unit m/s and can be presented as the output of the subsystem via an output port.

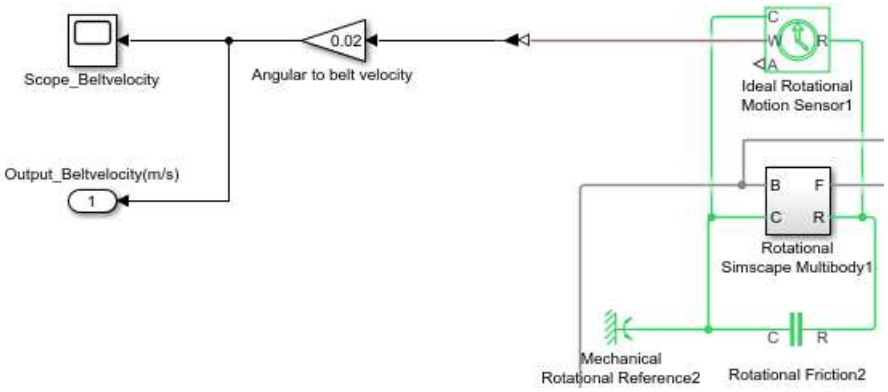


Figure 24: Measuring belt velocity

7.2 MODEL OVERVIEW

The completed model is shown in figure 25. This model is compressed into a subsystem with only the manipulated variable (MV) as input and the measured variable (MO) as output. Where MV equals the input voltage of the motor and MO equals the belt speed.

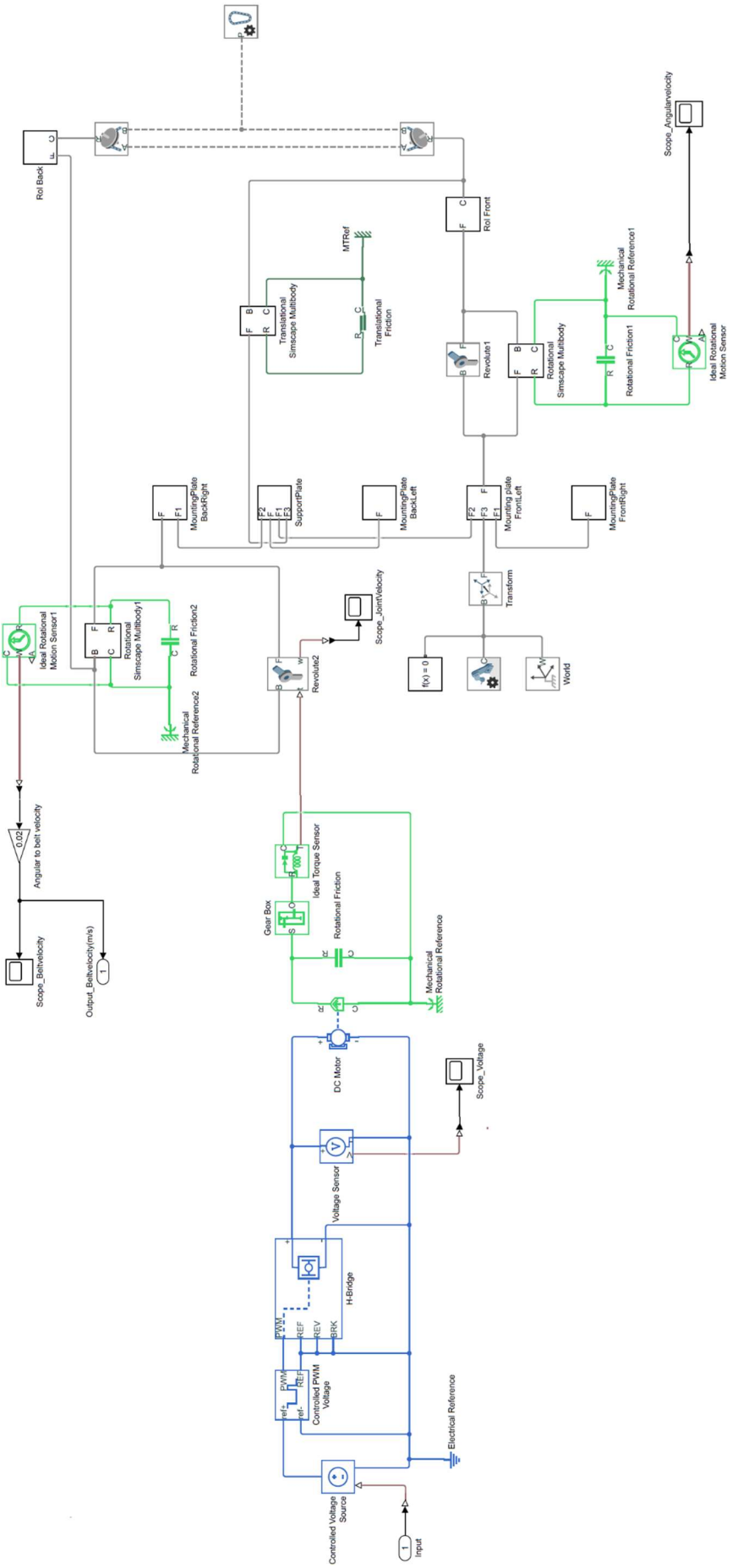


Figure 25: Complete Simulink Multibody model of the conveyor belt

8 CONTROL MECHANISMS

This chapter describes the different control mechanisms for the conveyor belt. PI/PID and MPC controllers are designed. These controllers will perform two different controls: speed control and position control. Position control is the main object of the conveyor belt while the speed control is auxiliary. Speed control can easily eliminate speed variations so that position control becomes more accurate in, for example, a cascade control loop. In the literature review, the theory behind the PI/PID and MPC controllers is discussed. This chapter is therefore limited to the practical implementation of the controllers.

8.1 REFERENCE SIGNAL

Each control loop requires an input signal or a reference signal. This signal describes the behaviour the system should perform in the ideal case. The conveyor belt can be controlled by speed or position. Therefore, the reference signal for both variables must be provided.

The reference signal is constructed by creating the profile of the acceleration and integrating it. A signal is created that only accelerates linearly, decelerates linearly and keeps the acceleration constant. When this signal is then integrated, a smooth speed profile with an S-curve is realized. These linear accelerations are created by adding and subtracting the signals from different *Ramp blocks* in Simulink, as shown in figure 26. Note that in the Simulink model there is a *Transport Delay block* with a value of 2 seconds. This is to prevent the signal from accelerating immediately so that the behaviour of the controllers becomes more visible. Furthermore, there is a *Gain block* with a value of 0.08. This is a scaling of the signal to ensure that the hypothetical object remains on the conveyor belt. In other words, so that the total distance travelled does not exceed the length of the conveyor belt.

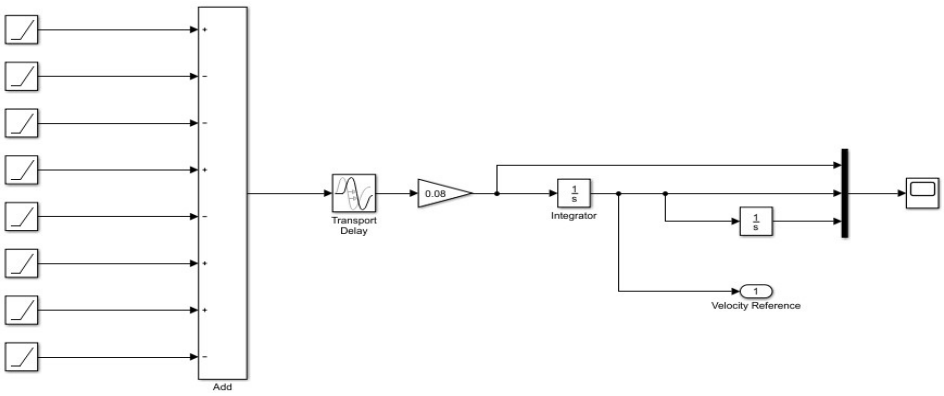


Figure 26: Simulink model of the reference signal

The idea behind the construction of this signal is explained by the first two *Ramp blocks*.
Set the start time of the second ramp one second later than the first and keep the slope the same. Then, subtract the second ramp signal from the first one. The result is shown in magenta in figure 27. There is a linear increase in the interval [0, 1]. But from 1 second onwards, the signal is constant.

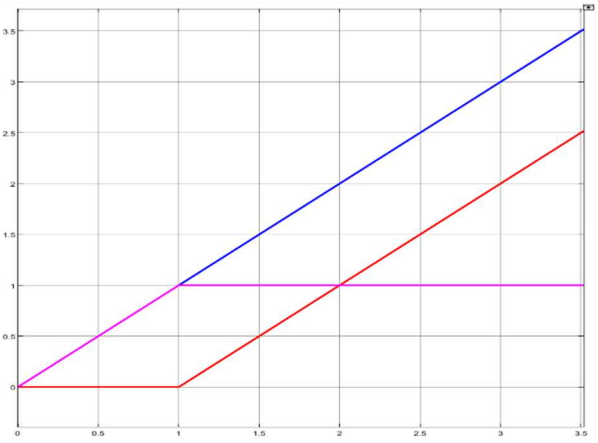


Figure 27: Subtraction of ramp signal blocks with Ramp1 (blue), Ramp2 (red) and result (magenta)

This technique was applied several times to get to the signal shown in green in figure 28. In the interval [2, 5], the conveyor belt accelerates. The shape of the acceleration profile results in a clean S-shaped velocity curve without harsh movements. Between 5 and 7 seconds, there is no acceleration and the velocity is constant at a maximum speed of 0.16 m/s or 16 cm/s. After 7 seconds, the conveyor belt brakes analogous to the acceleration curve. The red signal is the distance travelled on the conveyor belt. This signal is obtained by integrating the speed profile and is used as a reference for position control.

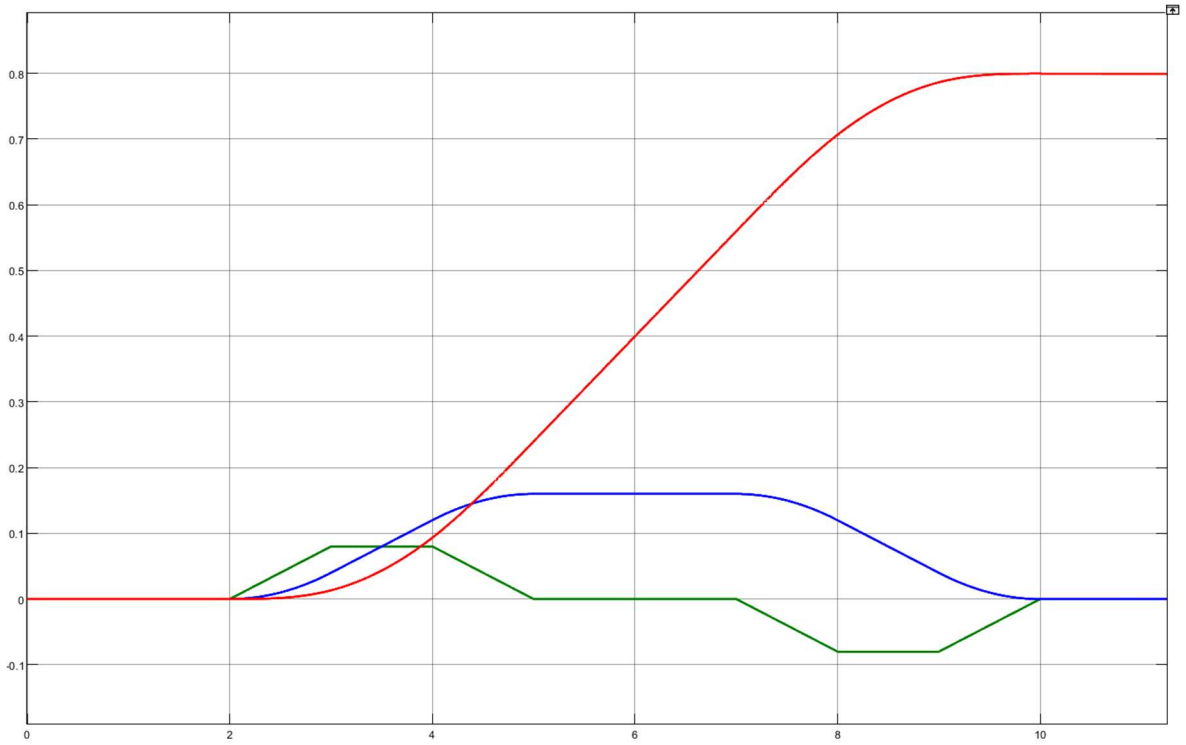


Figure 28: Acceleration (green), velocity (blue) and position (red) reference signal

8.2 MODEL LINEARIZATION

Linearization involves creating a linear approximation of a nonlinear system that is valid in a small region around the operating point. It is needed to design the PI/PID and the MPC control systems using classical design techniques. In this section, the conveyor belt model, further referred to as plant, is linearized [16].

First, the behaviour of the system is examined. This behaviour is shown in figure 29. Note that the belt speed is amplified for clarity and not to scale. When the voltage is gradually increased, the conveyor will also gradually accelerate. However, when a low voltage is applied to the system, the plant will show different behaviour, highlighted by the red circle. This deviation is caused by friction in the plant. At low input voltages, the motor does not deliver enough torque to overcome the static friction. When the input value exceeds about 2V, the system will show linear behaviour. Due to its behaviour at low voltages, the system is non-linear and must be linearized before controllers can be designed.

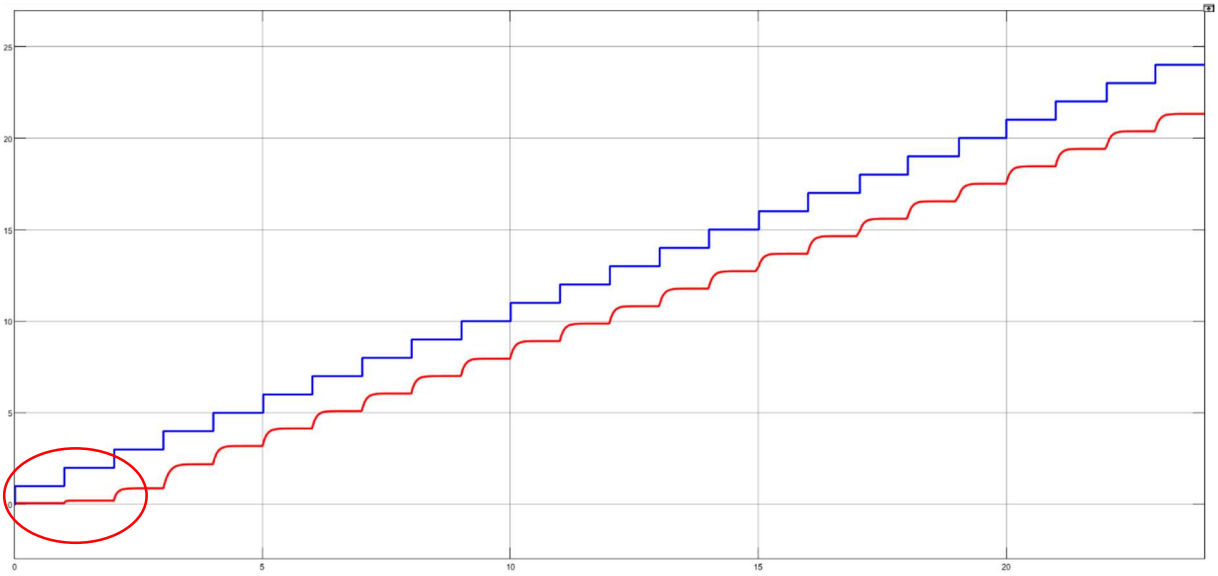


Figure 29: Belt speed response (red) to a step by step voltage increase (blue [V])

The linearization must have an operating point in the linear area in order to be a good approximation for the system. To ensure this operating point, a constant voltage of 10V is applied to the plant before the linearization procedure is performed. The linearization is done with the Linear Analysis Tool. This tool is accessed by right-clicking on the *Plant block* and selecting "Linearize Block". This action opens the window shown in figure 30.

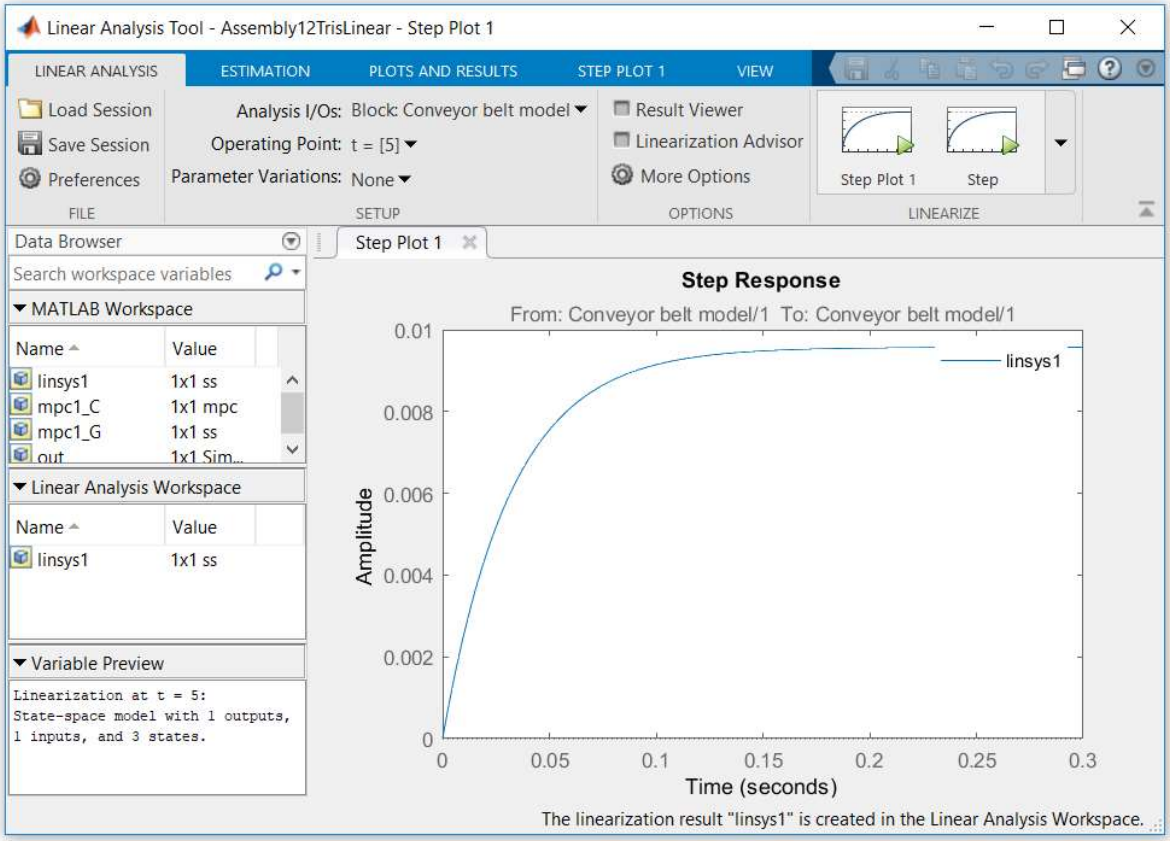


Figure 30: System Step Response at t=5 in the Linear Analysis Tool

A new state-space model (ss) is created and stored in the workspace under the name "linsys1". This state-space model contains variables specific to the "internal" state of the system. These describe the complete conveyor belt with its power supply and DC motor. This state-space model is used in the following sections to create the controllers.

8.3 PID

In this section, the PI/PID controllers for the plant will be designed. The *block PID Controller (2DOF)* is put in front of the plant. A random basic feedback control system using a 2-DOF PID controller is shown in figure 31.

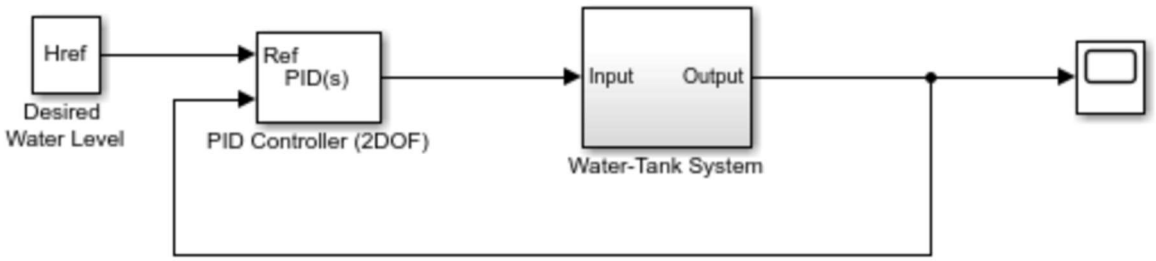


Figure 31: Basic feedback control system using a 2-DOF PID controller [17]

This block implements a continuous-time two-degree-of-freedom PID controller and has the option to choose between a PI, PID or PD controller. The output signal of this block is generated based on the difference between the reference signal and the measured system output. The transfer function of the *block PID Controller (2DOF)* is as follows [18]:

$$G_{PID-2DOF} = K_p \left[(b.r - y) + \frac{1}{\tau_i s}(r - y) + \frac{\tau_d s}{\frac{\tau_d}{N}s + 1}(c.r - y) \right] \tag{20}$$

The block computes a weighted difference signal for the proportional and derivative actions according to the setpoint weights (b and c) that are tuned. The block output is the sum of the proportional, integral and derivative actions on the respective difference signal, where each action is weighted according to the gain parameters P, I, and D. Also, a first-order pole filters the derivative action [17]. A control system has a degree of freedom which is defined as the number of closed-loop transfer functions that can be adjusted independently [19].

The first thing that happens after opening the PID tuner is linearizing the plant. After this it tunes the parameters of the PID controller. However, this results in an error message: “PID Tuner could not find an initial stabilizing controller”. This is due to the static friction that occurs in the conveyor belt. When the motor starts to work, the torque it produces is not sufficient to overcome the static friction. Because of this, a limitation has to be added to make sure that the lower limit of the input signal is not zero but equal to 2V. In section 8.2 is described that the system behaves linear above 2V. This adjustment can be made in de block parameters under output saturation. Next, click on apply and reopen the PID tuner. The PID tuner is now able to find an initial stabilizing controller and the parameters can be transferred to the PID controller by clicking update block.

By reviewing the values of P, I and D action, it is noticeable that the values of the P and I action are very large. This is due to the PID controller being tuned at near zero velocity. There, the process gain and time constant of the system is very small. Due to this, the tuner calculates very large values for both the proportional and integral gain. This is noticeable by applying a step to the open loop system. First a step from 0V to 1V which has a very low time constant and then a step from 10V to 11V. The result is different.

By running the program, it is noticeable that despite the high values of the P and I action the tuning works. This is because the dynamics in higher velocities are almost purely first order linear dynamics. It is hard to make that kind of systems unstable even with very large values of controller gains.

8.3.1 Speed control

In this section, the design of a PID controller for the belt speed is explained. Gain scheduling is an approach to control non-linear systems that use multiple linear controllers. Each of these controllers provides good control for a different operating point of the system. The parameters of all these individual controllers are stored in lookup tables. More information about the lookup tables can be found at the end of this section. To obtain these parameters from the tables, the following steps need to be executed. First, perform a step response experiment to get the linearized model parameters from the graphs. The transfer function model will be of the form $G(s) = K/(\tau s + 1)$ because there will be no dead time. In the transfer function, K is the steady state gain and τ is the time constant. The experiment can be performed using the model shown in figure 32. Running the model results in the response shown in figure 33.

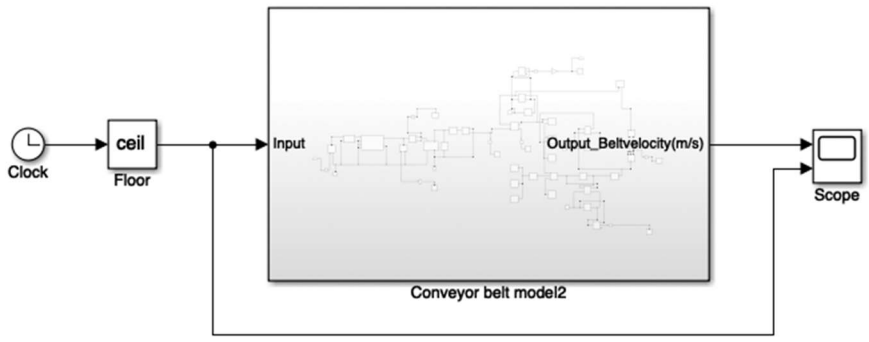


Figure 32: Model for step response experiment

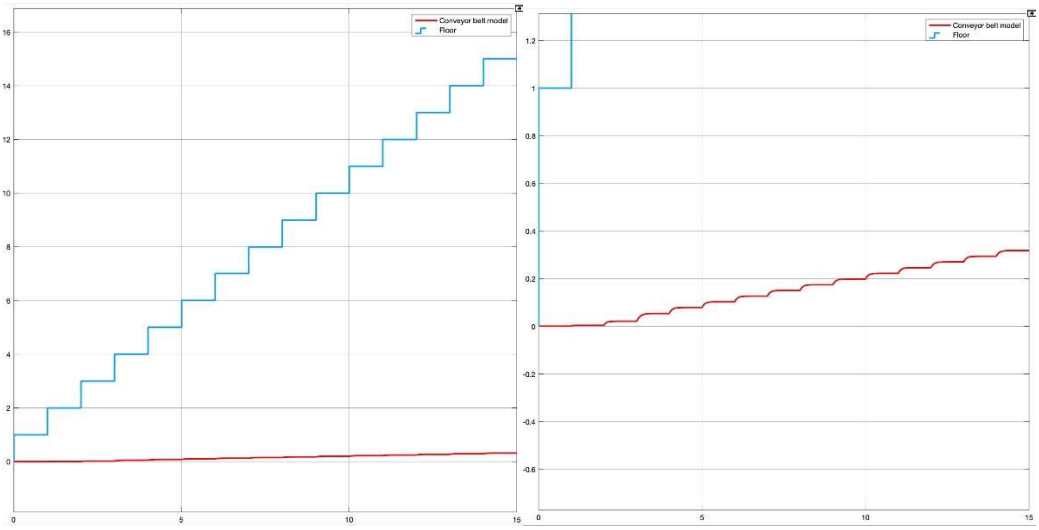


Figure 33: Step response experiment result

From the response shown in figure 33 the gain and time constant can be obtained for different operating points of the system. This is done by zooming in on the response of a step, as shown in figure 34.

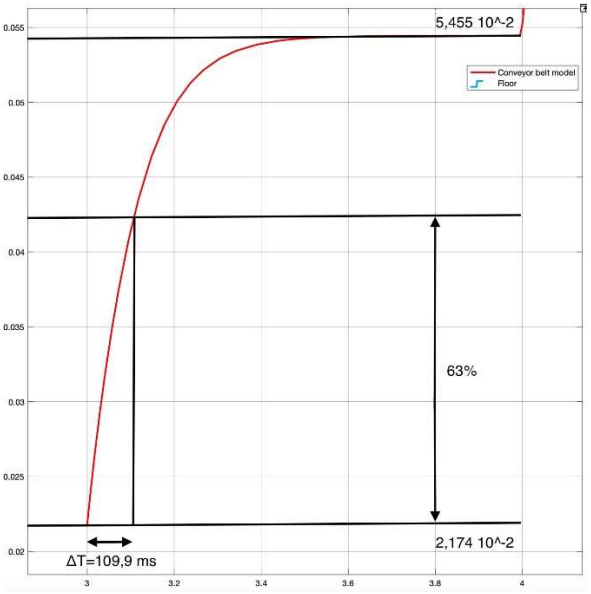


Figure 34: Determining gain and time constant

The time constant is equal to the time it takes to reach 63% of the step. The gain is the difference between the value when the step starts and the maximum value that is reached during that step. This process was repeated for steps 1, 2, 4, 6, 8, 10, 12, 14 and 16. Table 2 shows the time constant and gain for each step. Note that the variation on the measurements from step 6 and onwards is rather due to measurement errors, rounding errors and discretization errors than from differences within the system. A smaller number of controllers, for example three, would provide a similar system response.

Table 2: Time constant and gain from step response experiment

Step	Time constant (ms)	Gain
1	4,29	0,00121
2	24,30	0,00350
4	109,91	0,03271
6	82,96	0,02391
8	81,85	0,02389
10	82,06	0,02385
12	116,35	0,02389
14	119,69	0,02389
16	88,66	0,02389

With these values, the transfer functions of the model can be determined for each specific operating point that has been measured. With these transfer functions, a PID controller can be tuned for each plant model. Determining the transfer functions and the tune of the PID controllers is done by executing following lines of code:

```
ny = 1;
nu = 1;
nModels = 9;
s = tf('s');
sys = tf(zeros(ny,nu,nModels));
num = [0.00121; 0.00350; 0.03271; 0.02391; 0.02389; 0.02385; 0.02389;
0.02389; 0.02389];
den = [0.0043*s+1; 0.0243*s+1; 0.1099*s+1; 0.0830*s+1; 0.0818*s+1;
0.0821*s+1; 0.1163*s+1; 0.1197*s+1; 0.0887*s+1];
for k = 1:nModels
    sys(:,:,k) = num(k)/den(k);
end
step(sys)
Controllers = pidtune(sys,'pidf2');
Controllers
```

The values of the nine different PID controllers are now visible in the command window and are collected in the table 3.

Table 3: Parameters of controllers for lookup tables

Kp	Ki	Kd	Tf	b	c	Breakpoints
1110	514000	-0,173	0,00114	0,0303	0,22	1
382	31400	-0,337	0,00643	0,0303	0,22	2
40,9	744	-0,163	0,0291	0,0303	0,22	4
55,9	1350	-0,169	0,022	0,0303	0,22	6
56	1370	-0,166	0,0216	0,0303	0,22	8
56,1	1370	-0,167	0,0217	0,0303	0,22	10
56	962	-0,236	0,0308	0,0303	0,22	12
56	935	-0,243	0,0317	0,0303	0,22	14
56	1260	-0,18	0,0235	0,0303	0,22	16

For each parameter a lookup table must be created. But before this can be done, the parameters need to be imported in Matlab. Here, the data import feature of Matlab is used to import the Excel file. The import creates seven variables in Matlab with the name and values of the parameters in table 3.

As the variables have been created now, the gain-scheduled PID controller can be implemented. First, the source for the controller parameters needs to be set to external. Using external inputs allows the coefficients to vary. Double-click the *PID Controller (2DOF) block* to examine the configuration and if necessary, change the controller parameters source to external.

Next, lookup tables must be added for each parameter of the PID controllers. A 1-D Lookup Table block is used for each of the PID parameters. To see how the lookup tables are configured, double-click the *LookupTable_P block*. The block configuration is shown in figure 35.

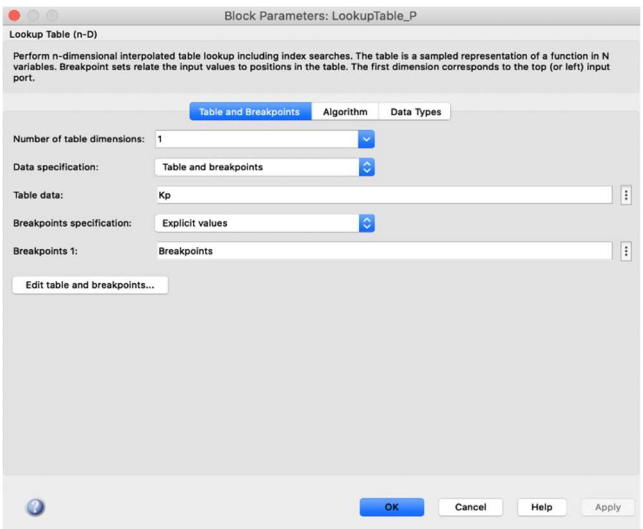


Figure 35: Block parameters LookupTable_P

The Table data contains the array of proportional coefficients Kp for each controller. Each entry in this array corresponds to an entry in the array Breakpoints that is entered in the Breakpoints 1 field. For concentration values that fall between the values of the variable Breakpoints, the lookup table block performs linear interpolation to determine the value of the proportional coefficient. The other lookup tables are set up in the same way but with the corresponding variable as Table data. Only the *LookupTable_N block* is slightly different.

The PID models in the controller's array express the derivative filter coefficient as a time constant Tf. However, the PID controller block expresses the derivative filter coefficient as the inverse constant N. Therefore, the *LookupTable_N block* must be configured to use the inverse value of each value in Tf. The configuration of the *LookupTable_N block* is visible in figure 36. The configuration of the model is finished. The complete model is shown in figure 37.

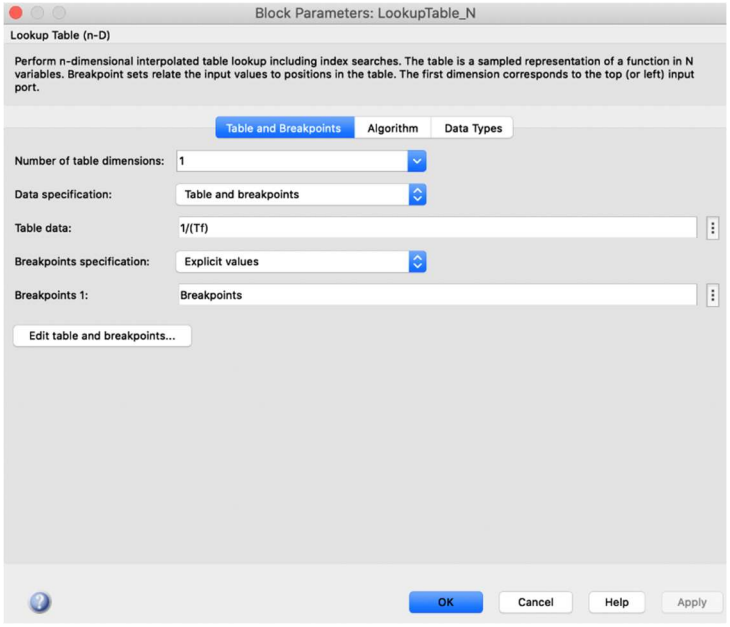


Figure 36: Block parameters LookupTable_N

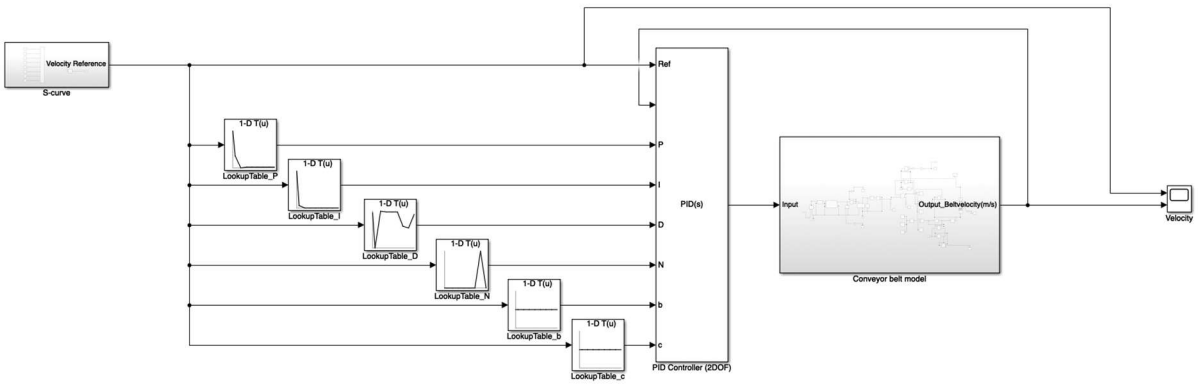


Figure 37: Simulink model of the PID with belt speed as measured variable

The controller is fully tuned and the behaviour of the system can be described. In figure 38, the velocity reference is shown in green and the system response in m/s is shown in red. It can be concluded that the system reacts quite well without overshoot and static errors. The system follows the reference signal almost perfectly. This is because a gain scheduling PID controller is used which greatly improves the accuracy of the controller and this can also be observed in the system response.

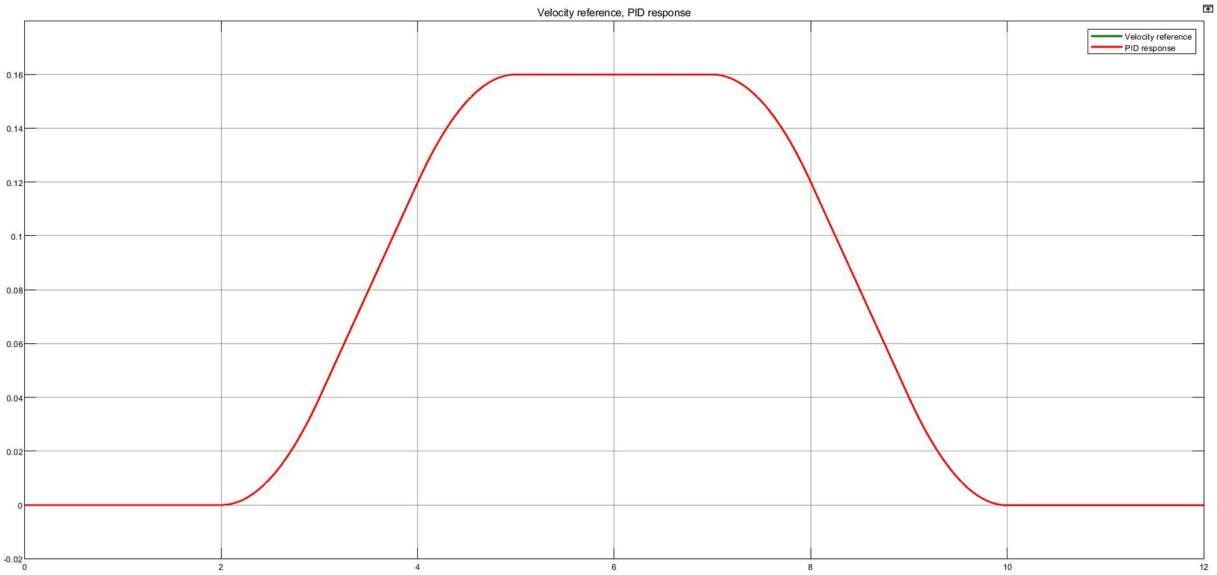


Figure 38: PID controlled system response with belt speed as measured variable

8.3.2 Position control

The position control of the plant is obtained with the model shown in figure 39. In this case, a cascade circuit is used where the inner loop is the speed control loop and the outer loop is the position control loop. In the general case of a cascade control, the master controller receives the error on the controlled variable as input signal. The output of the master controller serves as a set value for the slave controller which controls another variable. In motion control, the position of the conveyor belt could be achieved with a single PID controller. However, much more accurate control of the conveyor is achieved by sensing position and velocity, controlling each variable with its own loop [20], [21].

The already existing speed control of section 8.3.1 with its controller is used as the inner loop. This controller and its lookup tables is placed in a subsystem so that the model looks organized. An extra controller is added as a control mechanism for the outer loop position control. The reference signal of this PID controller is the *Position Reference block* as described in section 8.1. The feedback signal is the position of the belt and is obtained by integrating the belt speed which is the output of the plant. The output signal of the controller is used as the reference signal for the subsystem *PID for velocity control*.

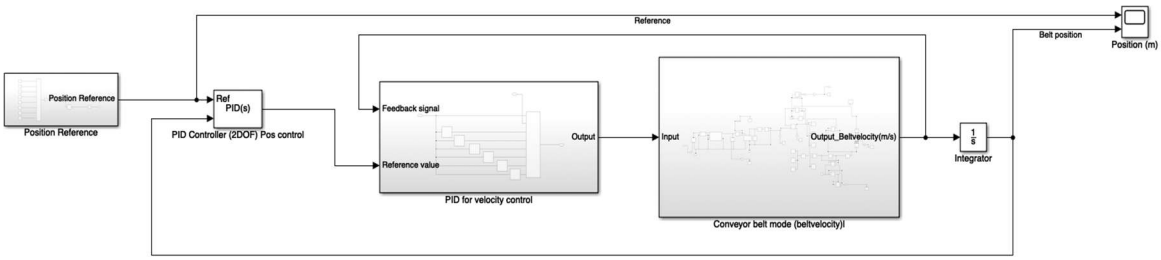


Figure 39: Simulink model of the PID with belt position as measured variable

Tuning the parameters for the PID controller is done in a different way as with the model for speed control. The plant models defined in section 8.3.1 are used with the corresponding controller. The model used to tune the PID controller is displayed in figure 40.

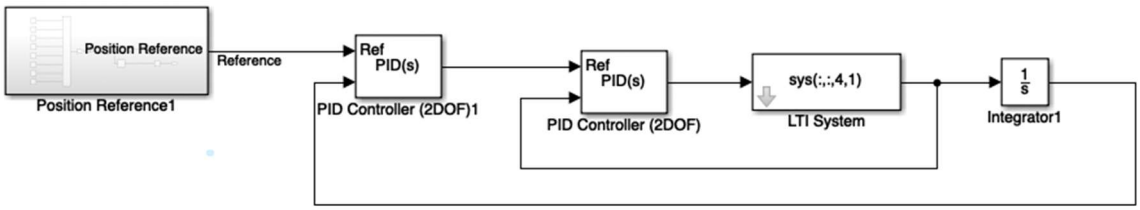


Figure 40: Simulink model used to tune the parameters of the PID controller for position control

The inner loop consists of an *LTI system block* and a *PID controller (2DOF) block*. In the *LTI system block,* the transfer function of plant model 4 can be inserted by entering the following in the LTI system variable: sys(:,:,4,1). The parameters for the gains of the *PID controller (2DOF) block* are those of the controller belonging to plant model 4. This is done by entering the following for the proportional gain: Controllers(:,:,4,1).Kp. The other parameters are defined in the same way. The controller for position control can now be tuned by pressing the Tuner button. Figure 41 shows the opened window with the values for the parameter gains and the step response.

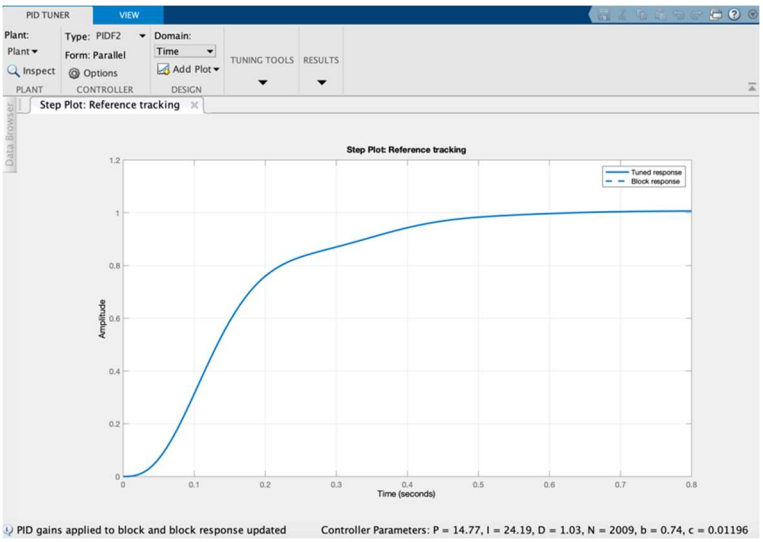


Figure 41: PID tuner

These values result in a system response shown in figure 42. The reference belt position is displayed in green and the system response is shown in red. The figure shows that the response of the system does not fully follow the reference signal. With a cascade circuit, the inner loop is always faster than the outer loop. This is also the case here. The outer loop gains are much smaller than the inner loop gains. Because of this, the outer loop is slower. Another reason for low gain values is the requirement for no overshoot. Overshoot in position would imply that the conveyor belt should move backwards, which is undesirable.

The time constant of the plant is very small at low voltages, therefore, the controller cannot react fast enough at that operating point. This causes a small deviation from the reference signal. It can be solved by using gain scheduling. This allows for higher values for the gains of the controller in the beginning and allows it to react faster.

However, the most important parameter of the operational conveyor belt is the final position of the product on the belt. At the end of the simulation, the variation on the end position is around 0.0004 m or 0.4 mm. This deviation is acceptable.

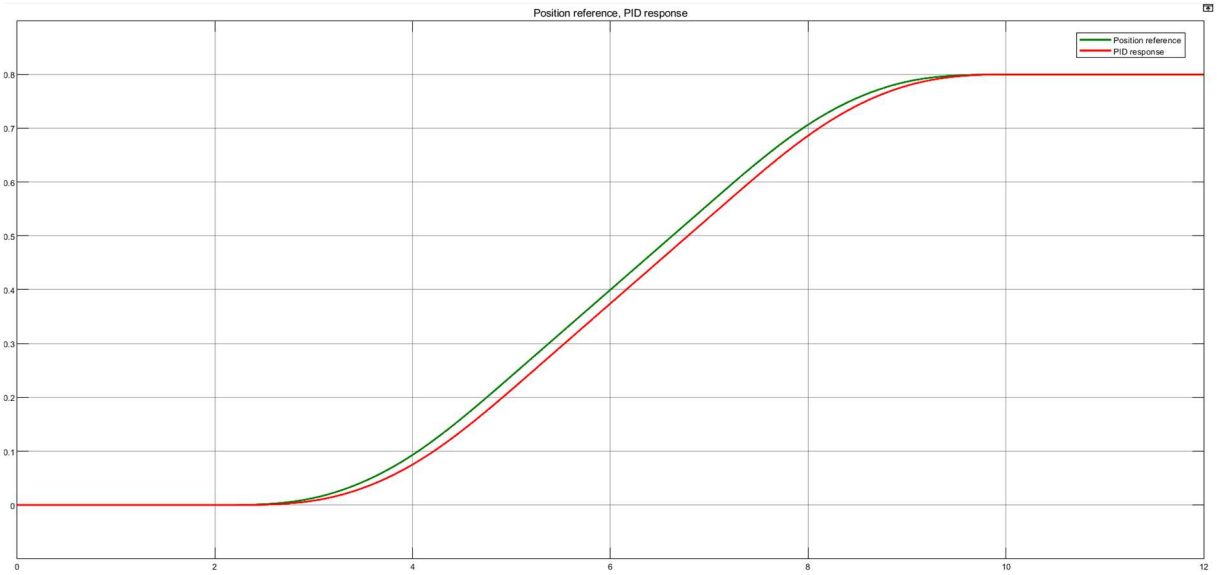


Figure 42: PID controlled system response with belt position as measured variable

8.4 MPC

In this section, the MPC controllers for the plant is designed. Due to the non-linear nature of the system at low voltages, the MPC cannot be designed in the conventional way. Normally one could put the MPC block in front of the model and design it by using the MPC Designer Tool. However, this method results in an error message: "Plant model cannot be a pure direct feed-through system without additional dynamics". The MPC must therefore be designed in a different way.

The MPC Designer Tool is accessed by typing "mpcDesigner" in the Matlab Command Window. Next, a plant model must be selected from the Matlab Workspace. Here, the linearized model in the form of a state-space model from section 8.2 is selected. When the plant model is defined and imported in the Designer Tool, the MPC controller is created. This controller is then exported by generating a Simulink model that uses the controller to control internal plant. The generated MPC block is then moved to the correct Simulink model with the reference and the plant. Now, the MPC controller is ready to be tuned for the best possible system response.

8.4.1 Design parameters

It is important to tune the MPC controllers so that the behaviour can be discussed. For this purpose, the right design parameters must be chosen. Choosing proper values for these parameters is important as they affect not only the controller performance but also the computational load of the MPC algorithm that solves the optimization problem at each time step.

First, the sample time is determined. This parameter sets the speed at which the controller executes the control algorithm. If it is too large, the controller will not be able to respond quickly enough to disturbances. If the sampling time is too small, the controller will be able to respond much faster to disturbances and setpoint changes, but this will cause an excessive computing load. To find the correct balance between performance and calculation effort, it is recommended to fit 10 to 20 samples within the rise time of the open-loop system response. This recommendation is shown graphically in figure 43 [22].

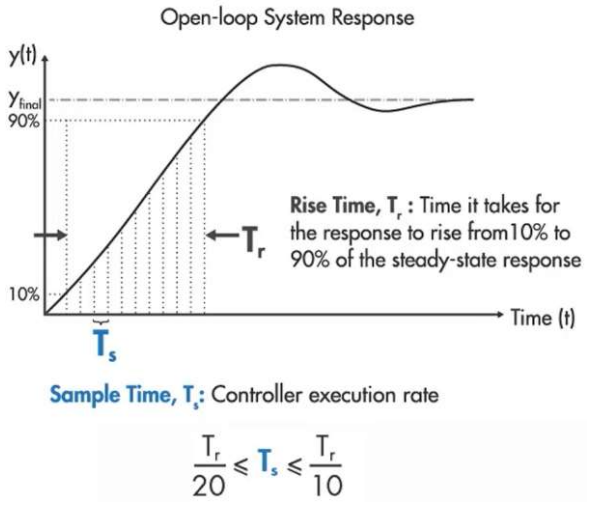


Figure 43: Graphical representation of the sample time recommendation [22]

The next design parameter is the prediction horizon. It is the amount of predicted future steps of the MPC and indicates how far the controller predicts the future. If it is too short, the system optimizes the behaviour based on the near future and thus ignores the long-term behaviour of the system. If the prediction horizon is too large, the closed loop system will be slow. Unexpected disturbances can make most of the predictions useless, wasting the computations. If the prediction horizon is short, disturbance effects are eliminated (too) quickly. For a large prediction horizon, elimination of disturbances is too slow. When the sample time is chosen according to the recommendation, the recommendation for choosing the prediction horizon is to have 20 to 30 samples covering the open-loop transient system response. This is shown graphically in figure 44 [22].

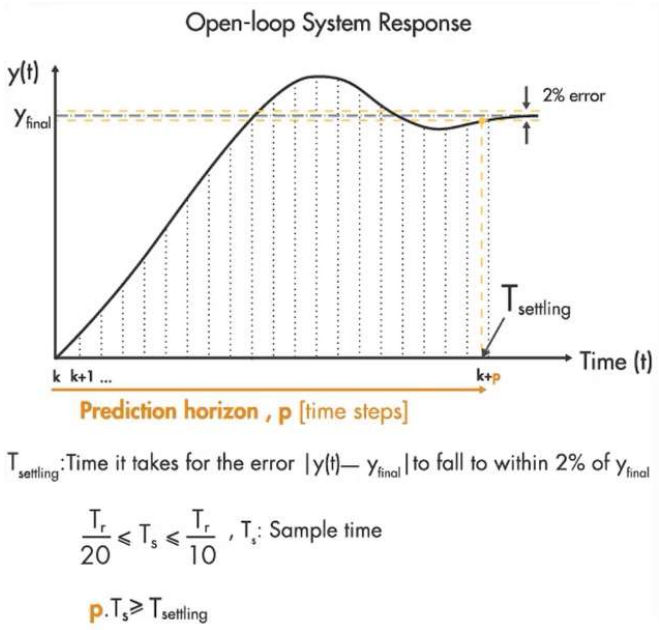


Figure 44: Graphical representation of the prediction horizon recommendation [22]

Finally, the control horizon is discussed. It is the number of control moves before the time step where the rest of the inputs are held constant. Each control move needs to be computed within the controller. So, a larger control horizon results in a heavy computational load. A control horizon of one might give the smallest number of calculations but the predictions might be less accurate and the performance is usually sluggish. Making the control horizon too big is also sub-optimal because only the first steps have a significant effect on the prediction. In addition, a large control horizon makes the controller too active forcing to use control increment weights as a primary way to reduce control variable oscillations. Therefore, it is recommended to choose the control horizon 10 to 20% of the prediction horizon with a minimum of 2-3 steps. This is shown graphically in figure 45 [22].

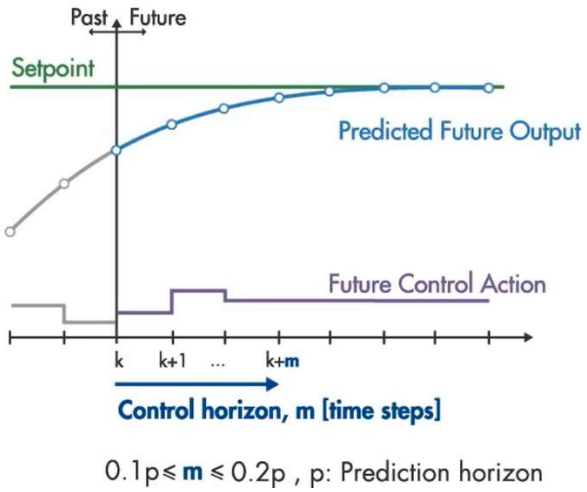


Figure 45: Graphical representation of the control horizon recommendation [22]

In the MPC Designer Tool, constraints can be set for the inputs and outputs. For example, a maximum input of 24V for the DC motor can be set. However, the controller would never apply this voltage to our application as this would result in the conveyor belt travelling at excessive speed. As a result, setting this constraint would have no effect on the behaviour of the system. No meaningful constraint could be imposed and, therefore, are not discussed further in the thesis.

8.4.2 Speed control

An MPC controller is designed for the belt speed as a measured variable. The implementation in Simulink is shown in figure 46.

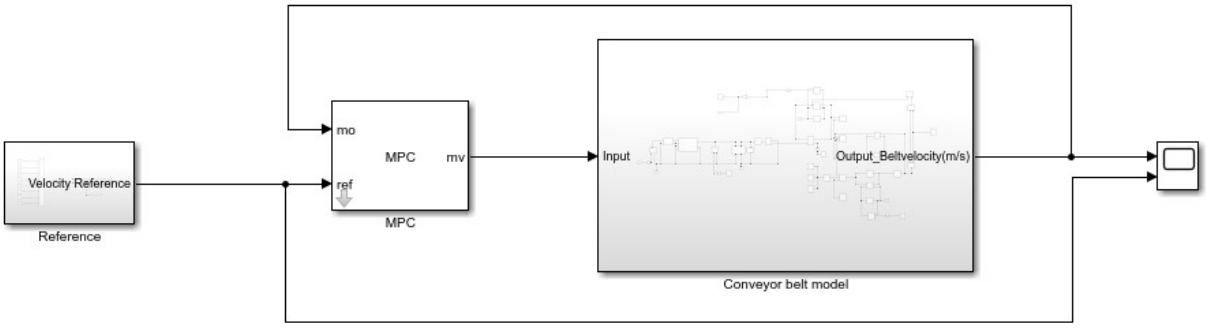


Figure 46: Simulink model of the MPC with belt speed as measured variable

The next step is to determine the design parameters. First, the correct step size is chosen. Here, the rise time of the open-loop system response is determined. To determine the rise time and settling time of the open loop system response, the command "stepinfo(linsys1)", where linsys1 is the linearized state-space model of the plant, is used. Execution of the command results in a rise time of 0.1762s and a settling time of 0.3138s. Taking the recommendations into account, a sample time

of 9ms, a prediction horizon of 20 times the sample time and a control horizon of 2 steps are chosen.

The controller is fully tuned and the behaviour of the system can be described. In figure 47, the velocity reference is shown in green and the system response is shown in blue. It can be concluded that the system reacts quite well without overshoot and static errors. However, it is noticeable that in first acceleration at $t = 2s$, the system is lagging. This slow start-up is due to the static friction of the model. Because the model is linearized at 10V, this part can't be modelled properly. Possible solutions for this problem are discussed in section 8.4.4.

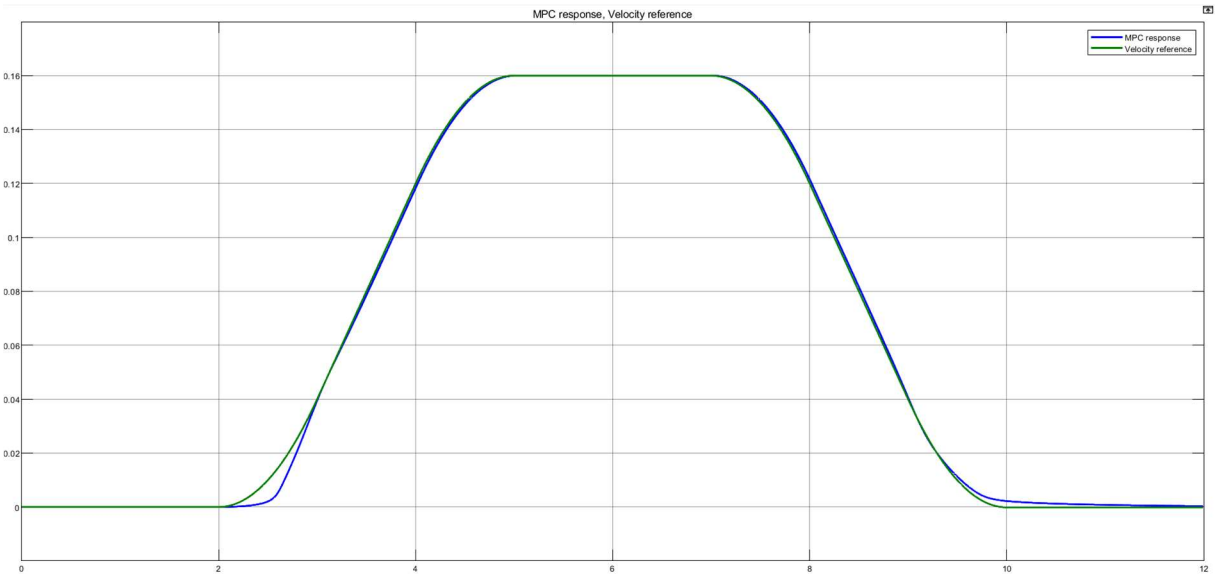


Figure 47: MPC controlled system response with belt speed as measured variable

8.4.3 Position control

An MPC controller is designed for the belt position as a measured variable. The implementation in Simulink is shown in figure 48. The position reference is the integrated signal of the belt velocity reference. Similarly, the conveyor belt model has now the belt position as output by adding an integrator to the output signal. Note that these adaptions are made inside the subsystems and are not observable in the figure.

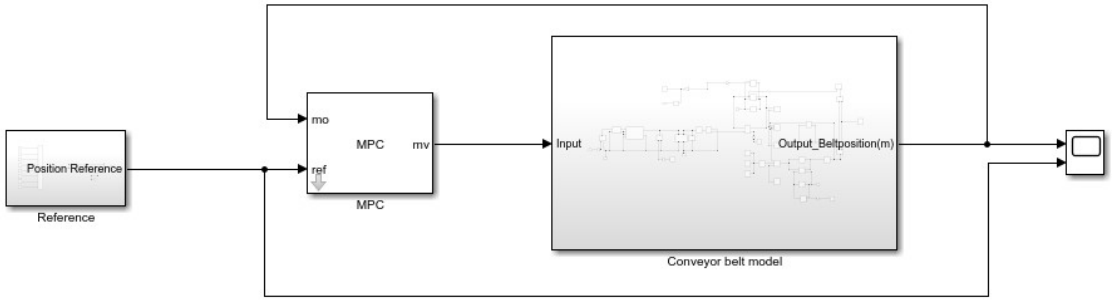


Figure 48: Simulink model of the MPC with belt position as measured variable

Similar to the belt speed, the design parameters must be chosen. A gain, a sample time of 9ms, a prediction horizon of 20 times the sample time and a control horizon of 2 steps are chosen. However, the initial response of the system was far too slow. Using the Performance Tuning sliders in the MPC Designer Tool, the closed-loop performance was made more aggressive and the state estimation was made faster. These changes result in a system response shown in figure 49. In this figure, the reference belt position is displayed in green and the system response is shown in blue. It can be concluded that the system response is satisfactory. The most important parameter of the operational conveyor belt is the final position of the product on the belt. At the end of the simulation, the variation on the end position is around 0.0004 m or 0.4 mm. This deviation is acceptable.

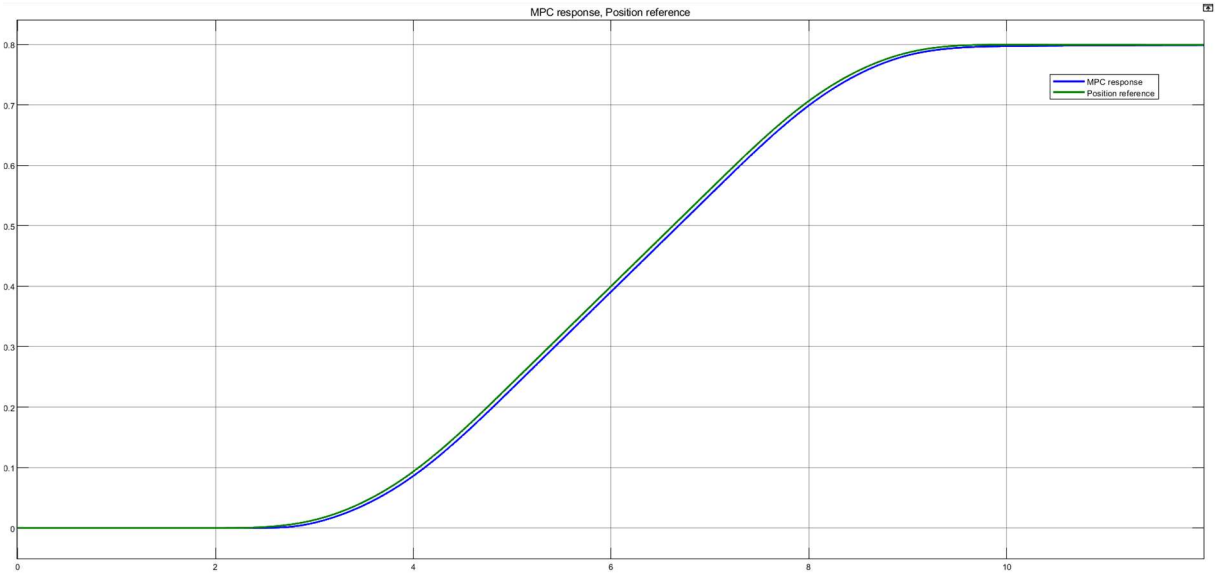


Figure 49: MPC controlled system response with belt position as measured variable

8.4.4 Possible improvements

As can be seen in the system response of figure 47, there is still room for improvement. Slow start-up is due to static friction. The model at 10 V cannot describe that part properly. It is possible to design the MPC at a lower voltage but then the deviation at the higher voltages will be larger.

A possible solution to this problem is the use of multiple MPC controllers. For example, a controller for low voltages and a controller for high voltages. An additional selection algorithm still needs to be designed to determine which controller is used. This control technique is known as Gain-Scheduled MPC. A second solution is to linearize the model in real time each time the operating point changes. This is only possible if the optimization structure remains fixed for the entire operating area and the constraints are the same everywhere. This method is known as Adaptive MPC. The described methods are possible improvements to the project but are not discussed further in this thesis.

9 COMPARISON CONTROL MECHANISMS

In the previous chapter, controllers are designed for speed and position control of the conveyor belt. This chapter describes the advantages and disadvantages of the different controllers.

9.1 SPEED CONTROL

A PID and an MPC controller are designed for the belt speed as a measured variable in the previous chapter. In this section, their behaviour is compared to each other. In figure 50, the responses are shown on a single scope image to make a good comparison. The velocity reference is displayed in black, the MPC system response in blue and the PID response in red.

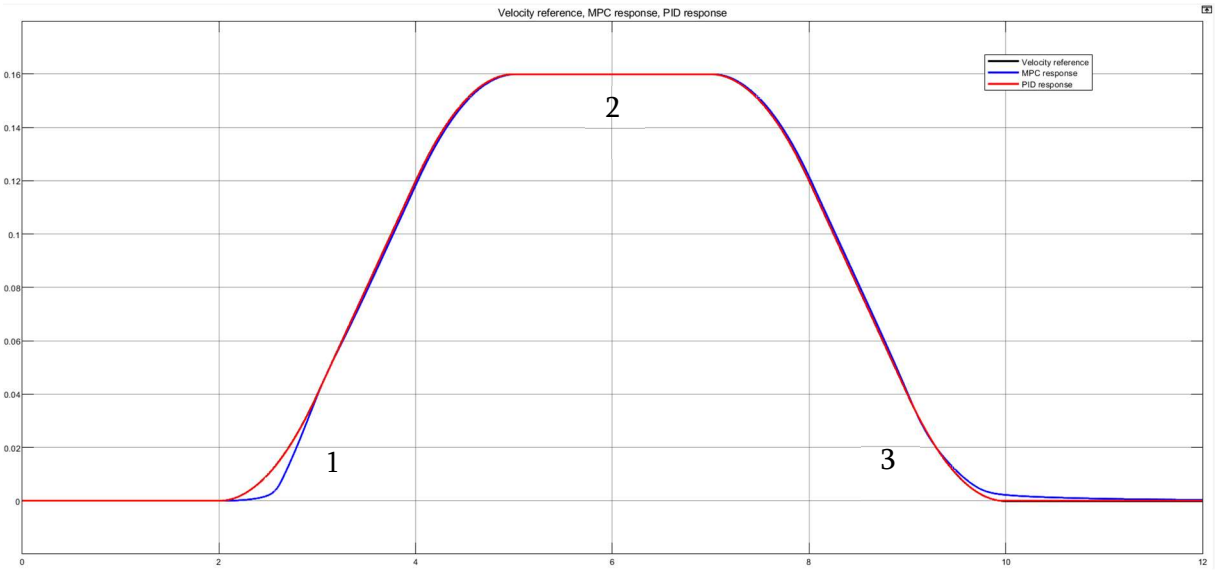


Figure 50: Comparison of MPC response and PID response with belt speed as measured variable

The PID controller follows the reference signal so closely that the signals seem to overlap. The MPC, on the other hand, has a slow start-up. This is even clearer in figure 51, the zoomed image of region 1. This slow start-up is due to the static friction of the model. Because the model is linearized at 10V, this part can't be modelled properly.

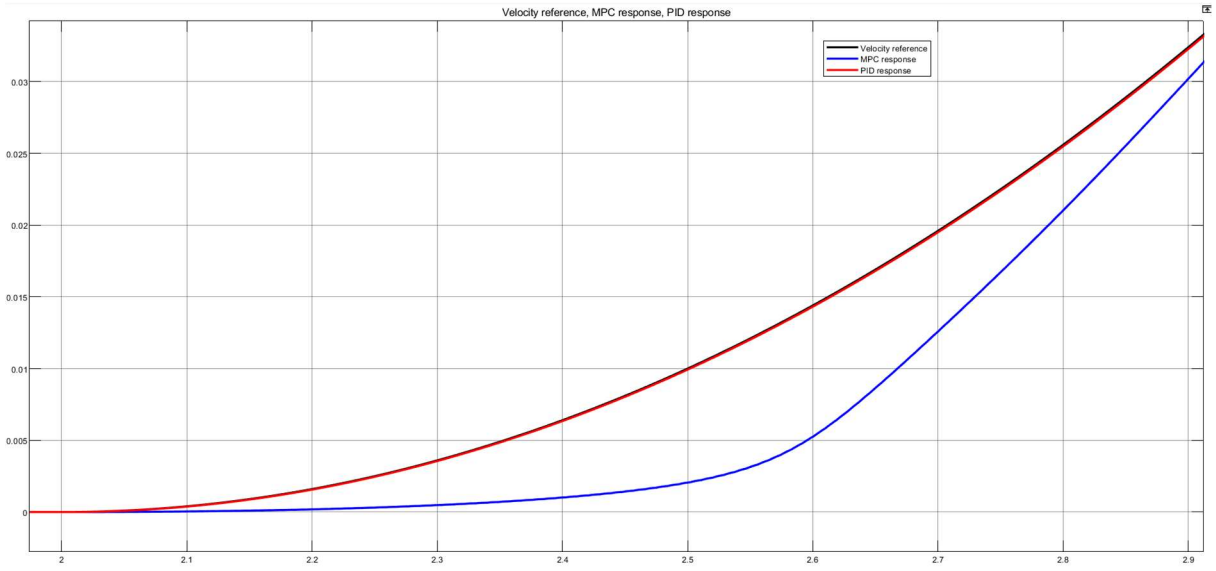


Figure 51: Zoom of region 1 of comparison of MPC response and PID response with belt speed as measured variable

Figure 52 gives a zoomed-in view of the response around the maximum speed. It is noticeable that the PID has a very accurate response. The response of the MPC in this region is significantly better than the response at low voltages. However, the response of the MPC is still slower than the PID.

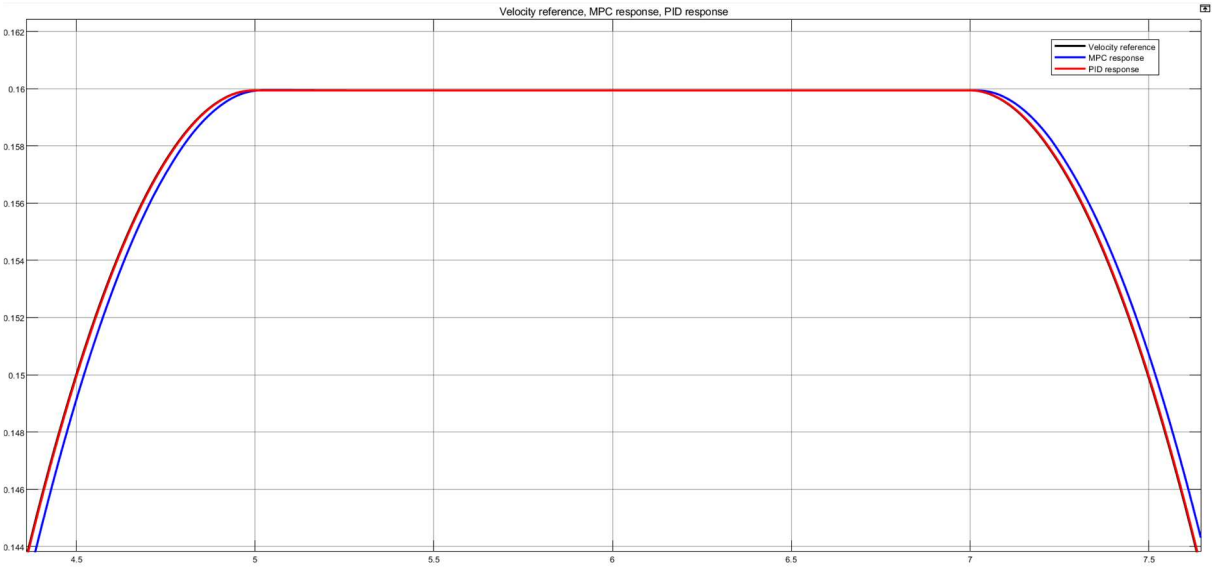


Figure 52: Zoom of region 2 of comparison of MPC response and PID response with belt speed as measured variable

Figure 53 shows a zoomed-in image of the response at the endpoint. Analogous to the start-up behaviour, the MPC has problems with control at low voltages. The PID is the most accurate controller in this case.

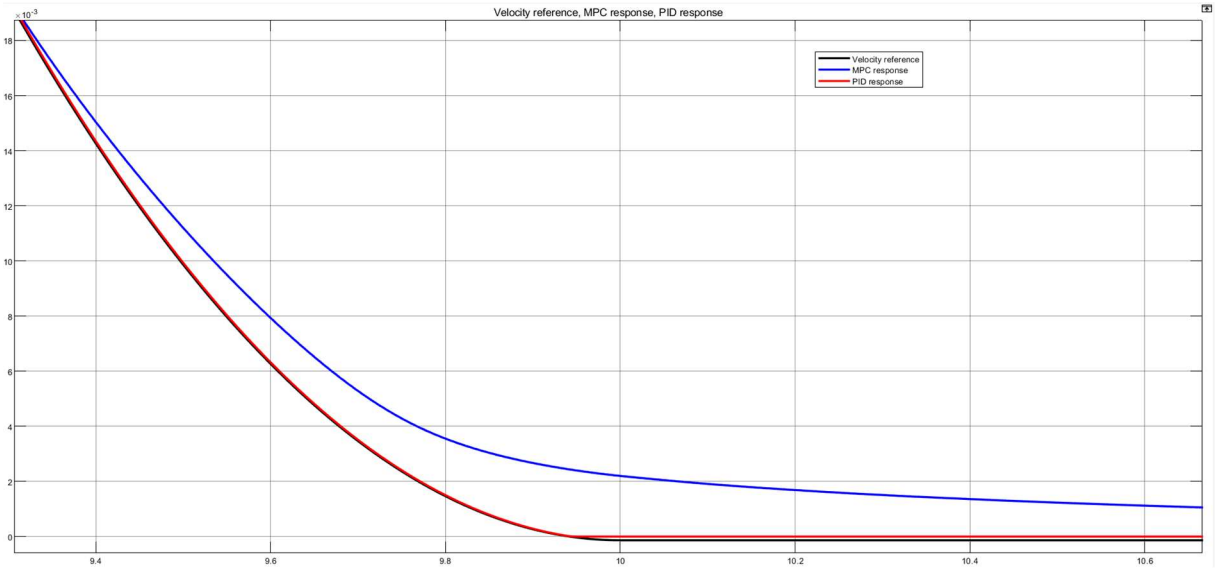


Figure 53: Zoom of region 3 of comparison of MPC response and PID response with belt speed as measured variable

For the conveyor application with speed control, it can be concluded that gain scheduled PID performs better than a single MPC. The main advantage of the gain scheduled PID is that there are different linearizations at different operating points. This results in an overall accurate response. The MPC is designed for a linearized conveyor belt model at 10V, resulting in deviations at low voltages. However, there is also a gain scheduled MPC as discussed in section 8.4.4 but it is not designed in this thesis.

The MPC also has advantages over the PID but not for this application. It is known that the PID controller is very strong for Single Input Single Output systems, like a conveyor belt with one voltage input and one belt speed output. When multiple variables are controlled in a Multiple Input Multiple Output system, the behaviour of the PID controller is less optimal. This is where the power of the MPC lies, because it is a multi-variable controller that takes the interactions between the different variables into account. However, an MPC can outperform PID in a SISO system if the dynamics are more complex. For example, a system with long dead times, right hand side zeros or otherwise complex dynamics.

9.2 POSITION CONTROL

A PID and an MPC controller are designed for the belt position as a measured variable in the previous chapter. In this section, their behaviour is compared with each other. In figure 54, the responses are shown on a single image to make a good comparison. The velocity reference is displayed in black, the MPC system response in blue and the PID response in red.

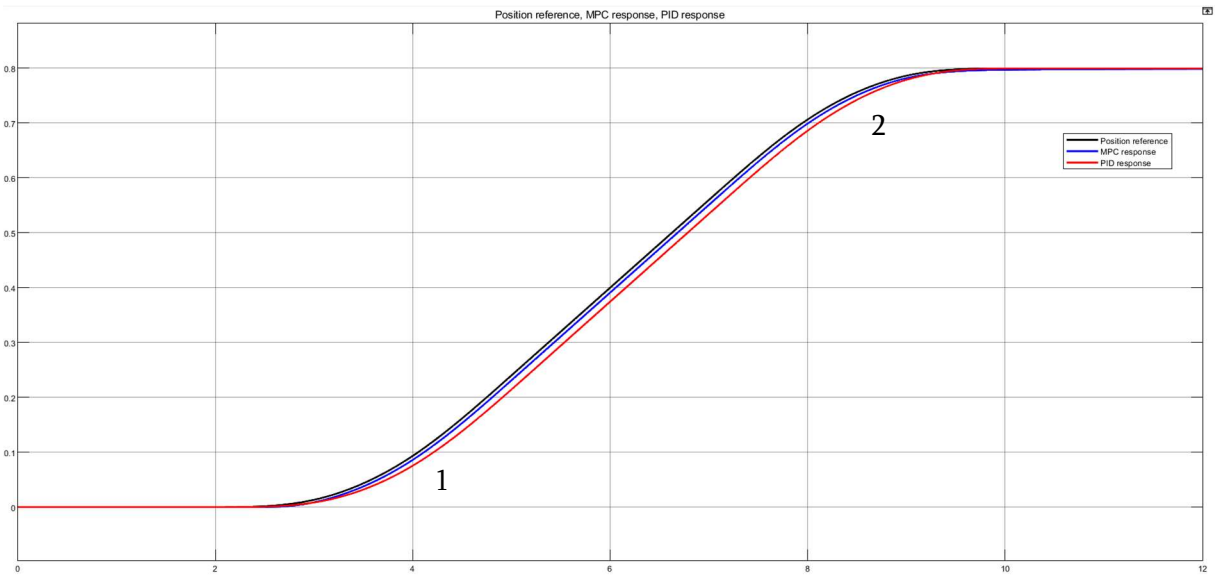


Figure 54: Comparison of MPC response and PID response with belt position as measured variable

The figure shows that the response of both systems does not fully follow the reference signal. This is shown more clearly in figure 55. For PID control, as mentioned in section 8.3.2, this issue is caused by the outer loop of the cascade circuit. The outer loop gains are much smaller than the inner loop gains. Because of this, the outer loop is slower. The time constant of the plant is very small at low voltages, therefore, the controller cannot react fast enough at that operating point. This causes a small deviation from the reference signal. The MPC is designed for a linearized conveyor belt model at 10V, resulting in deviations at low voltages. The start-up is slower than the PID response. However, the signals cross which signifies that the MPC is closer to the reference position.

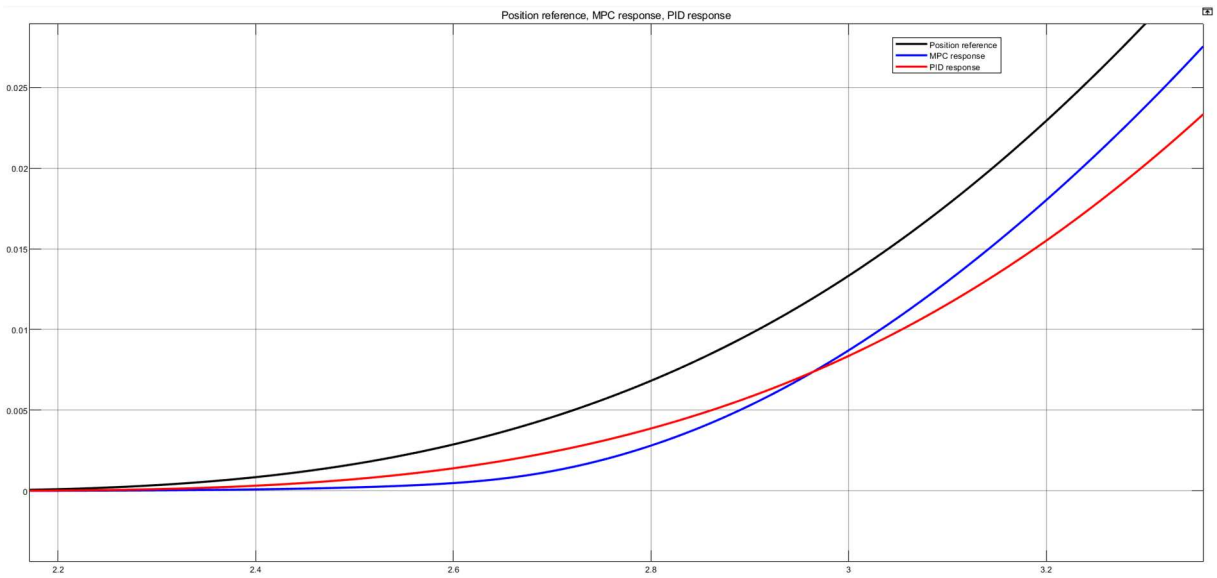


Figure 55: Zoom of region 1 of comparison of MPC response and PID response with belt position as measured variable

At the end position, shown in figure 56, both MPC and PID signals cross again. This implies that the MPC slows down faster so the PID reaches the end position first.

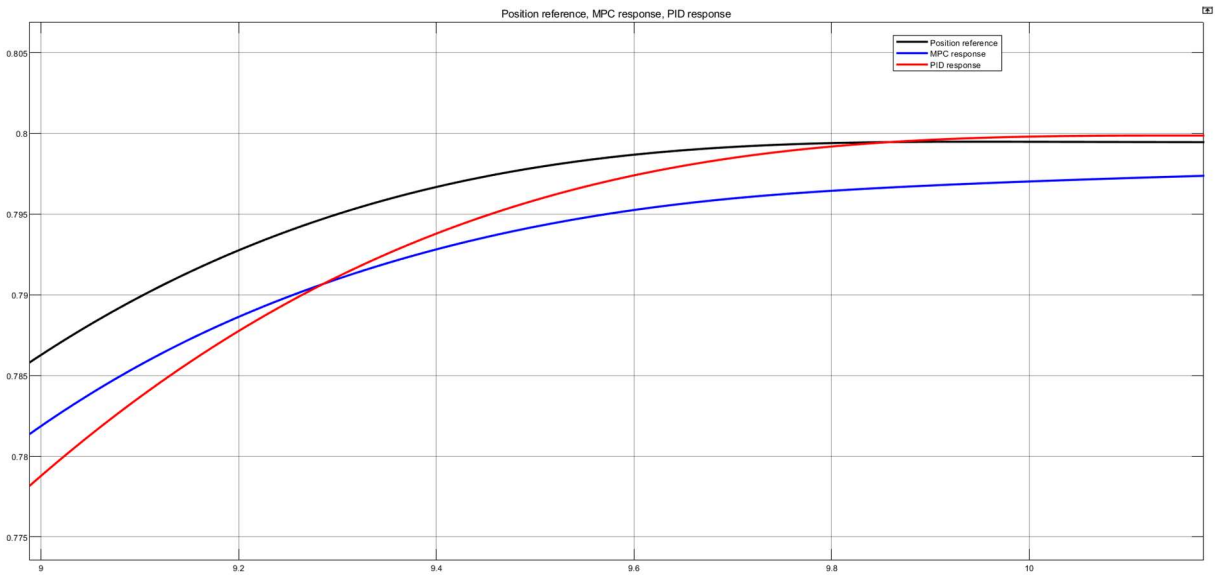


Figure 56: Zoom of region 2 of comparison of MPC response and PID response with belt position as measured variable

For the conveyor application with position control it can be concluded that both PID and MPC perform equally good. The PID starts faster than the MPC and reaches the final position first. The MPC on the other hand is overall closer to the reference position. The deviation on the final position is for both controllers acceptable. Both controllers can be improved by using multiple operation points for linearization as discussed previously. Gain scheduled MPC is expected to outperform the designed controllers for the conveyor belt. This is expected because it provides a solution for the non-linear dynamics of the system. However, for PID control, the position control is built as a cascade around the speed control. This means that the inner loop is already linearized due to the feedback. When the model is already linear, gain scheduling will not improve system behaviour.

10 CONCLUSION

In conclusion, this thesis investigated the toolchain from CAD to Simulink Multibody to real-time control. It tried to examine the identification of Simulink model parameters and it compared a PID controller with an MPC controller applied to a DC-powered conveyor belt.

The conversion from an Onshape model to a Simulink Multibody model is straightforward. If the CAD model is assembled properly without any geometrical errors and with correct degrees of freedom, the Simulink model will be generated correctly. The design process is accelerated by building the graphical model in a familiar CAD environment.

Because of the COVID-19 outbreak, practical tests on the conveyor belt were not carried out. This made it impossible to match the parameters with test values. No statements can be made about the complexity of this operation. The parameters were estimated in order to obtain plausible results.

Simulations have shown that for the conveyor application, a gain scheduled PID controller performs better than a single MPC controller at speed control. In position control, the PID and the MPC are equivalent. Here, both responses show deviations from the reference. This behaviour can be improved by using gain scheduled controllers. Note that these controllers have only been compared by simulations and the controllers have not been practically tested.

In future studies, gain scheduled controllers can be compared by means of simulations and practical testing with possible addition of disturbances.

REFERENCES

[1] Y. F. Liu, J. Li, Z. M. Zhang, X. H. Hu, and W. J. Zhang, 'Experimental comparison of five friction models on the same test-bed of the micro stick-slip motion system', *Mech. Sci.*, vol. 6, no. 1, pp. 15–28, 2015.

[2] A. Selezneva, 'Modeling and Synthesis of Tracking Control for the Belt Drive System', pp. 34–39, 2007.

[3] V. Van Geffen, 'A study of friction models and friction Compensation', *A study Frict. Model. Frict. Compens.*, pp. 1–24, 2009.

[4] J. Baeten, 'De klassieke regelaars', pp. 1–17, 2000.

[5] W. Chen and X. Li, 'Model predictive control based on reduced order models applied to belt conveyor system', *ISA Trans.*, vol. 65, pp. 350–360, 2016.

[6] S. Sahoo, B. Subudhi, and G. Panda, 'Optimal speed control of DC motor using linear quadratic regulator and model predictive control', *2015 Int. Conf. Energy, Power Environ. Towar. Sustain. Growth, ICEPE 2015*, vol. 55, no. 2, pp. 1–5, 2016.

[7] M. S. Gaya *et al.*, 'Enhanced pid vs model predictive control applied to bldc motor', in *IOP Conference Series: Materials Science and Engineering*, 2018, vol. 303, no. 1.

[8] Y. Jiang, C. Yang, J. Na, G. Li, Y. Li, and J. Zhong, 'A brief review of neural networks based learning and control and their applications for robots', *Complexity*, vol. 2017, 2017.

[9] H. Kim and C.-S. Lin, 'Use of adaptive resolution for better CMAC learning', in *Proceedings of Third International Conference on Signal Processing (ICSP'96)*, 2003, vol. 2, no. 10, pp. 517–522.

[10] 'Series RH158 | Micro Motors'. [Online]. Available: https://www.micromotors.eu/en/gear-motors/series-rh158/. [Accessed: 16-May-2020].

[11] 'Onshape Import - MATLAB & Simulink - MathWorks Benelux'. [Online]. Available: https://nl.mathworks.com/help/physmod/sm/ug/onshape-import.html. [Accessed: 16-May-2020].

[12] 'Ideal voltage source driven by input signal - MATLAB - MathWorks Nordic'. [Online]. Available: https://se.mathworks.com/help/physmod/simscape/ref/controlledvoltagesource.html. [Accessed: 03-Jun-2020].

[13] 'DC Motor Control - MATLAB & Simulink Example - MathWorks Nordic'. [Online]. Available: https://se.mathworks.com/help/physmod/sps/examples/pwm-controlled-dc-motor.html. [Accessed: 03-Jun-2020].

[14] MathWorks, 'Friction in contact between moving bodies - MATLAB - MathWorks Nordic', *MathWorks Documentation*, 2016. [Online]. Available: https://se.mathworks.com/help/physmod/simscape/ref/rotationalfriction.html. [Accessed: 03-Jun-2020].

[15] 'Motion sensor in mechanical rotational systems - MATLAB - MathWorks Nordic'.

[Online]. Available: https://se.mathworks.com/help/physmod/simscape/ref/idealrotationalmotionsensor.html. [Accessed: 03-Jun-2020].

[16] M. & Simulink, 'Linearization for Model Analysis and Control Design'. [Online]. Available: https://se.mathworks.com/discovery/linearization.html?.mathworks.com. [Accessed: 03-Jun-2020].

[17] 'Continuous-time or discrete-time two-degree-of-freedom PID controller - Simulink - MathWorks Nordic'. [Online]. Available: https://se.mathworks.com/help/simulink/slref/pidcontroller2dof.html. [Accessed: 29-May-2020].

[18] 'Two-Degree-of-Freedom PID Controllers - MATLAB & Simulink - MathWorks Nordic'. [Online]. Available: https://se.mathworks.com/help/control/ug/two-degree-of-freedom-2-dof-pid-controllers.html. [Accessed: 04-Jun-2020].

[19] M. Araki and H. Taguchi, 'Two-Degree-of-Freedom PID Controllers', 2003.

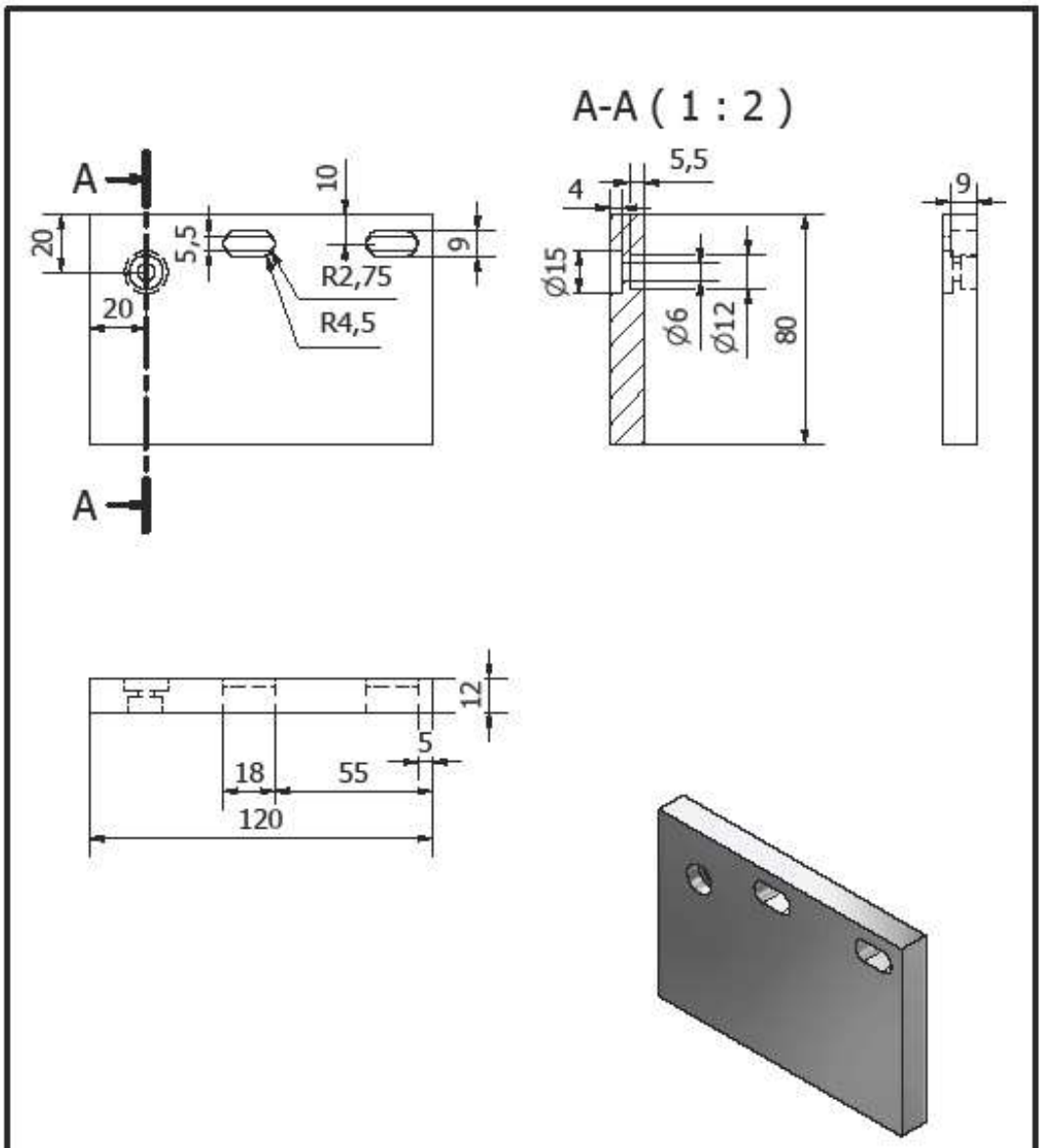
[20] 'Cascade Control | Basic Process Control Strategies and Control System Configurations | Automation Textbook'. [Online]. Available: https://control.com/textbook/basic-process-control-strategies/cascade-control/. [Accessed: 04-Jun-2020].

[21] 'Designing Cascade Control System with PI Controllers - MATLAB & Simulink - MathWorks Nordic'. [Online]. Available: https://se.mathworks.com/help/control/ug/designing-cascade-control-system-with-pi-controllers.html. [Accessed: 04-Jun-2020].

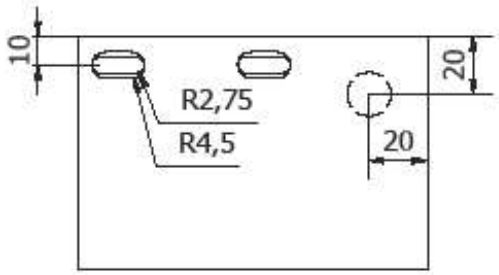
[22] 'Understanding Model Predictive Control, Part 3: MPC Design Parameters Video - MATLAB & Simulink'. [Online]. Available: https://se.mathworks.com/videos/understanding-model-predictive-control-part-3-mpc-design-parameters-1530607670393.html. [Accessed: 29-May-2020].

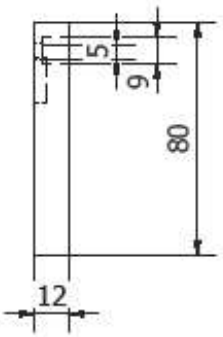
LIST OF ATTACHMENTS

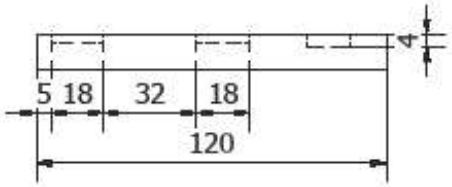
Attachment A: Drawings parts conveyor belt

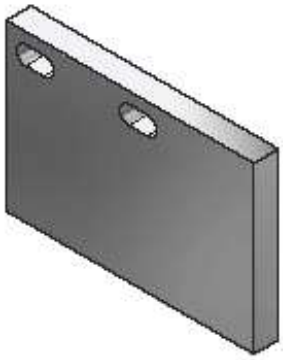


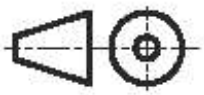
Materiaal: Aluminum		Ruwmaten: 120 x 80 x 12	
Norm:		Behandeling:	
Benaming: Mounting plate 1		Tek. nr:	Aantal: 2
		Stuk nr: 1	Sam. nr:
UHASSELT	Tekenaar: Ruben Berden	Formaat: A4 / Schaal: 1:2	
KU LEUVEN	Groep:	Datum: 10/05/2020	

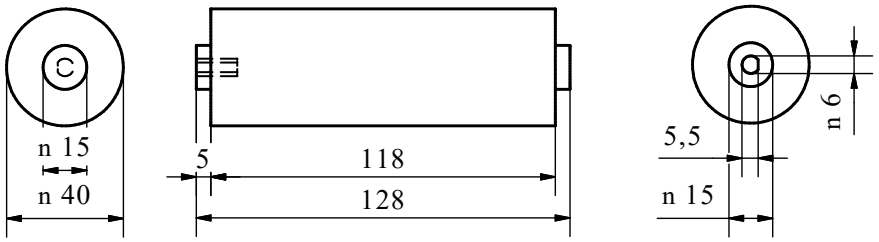


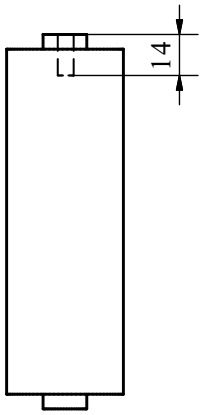


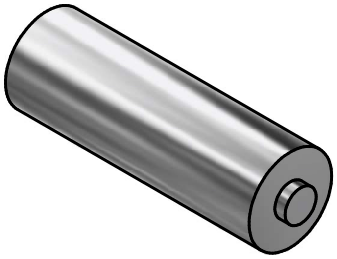


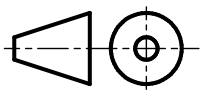


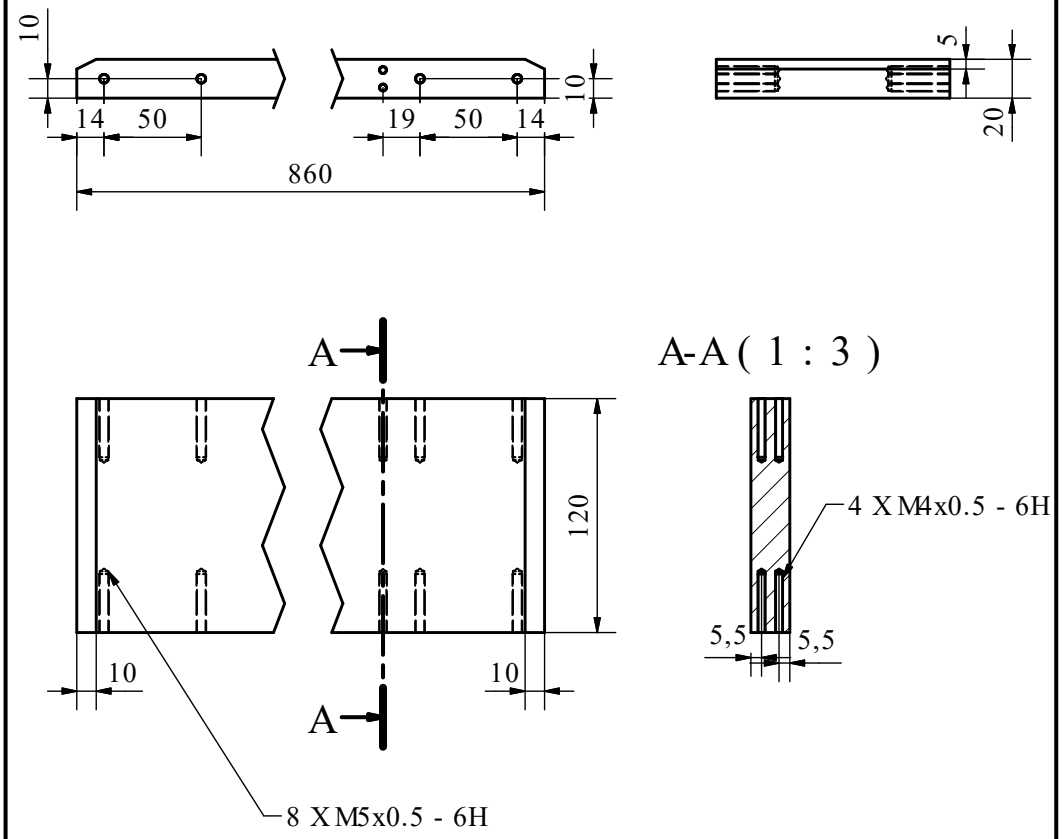
Materiaal: Aluminum		Ruwmaten: 120 x 80 x 12	
Norm:		Behandeling:	
Benaming: Mounting plate 2		Tek. nr: 2	Aantal: 2
		Stuk nr:	Sam. nr:
UHASSELT	Tekenaar: Ruben Berden	Formaat: A4 / Schaal: 1:2	
KU LEUVEN	Groep:	Datum: 10/05/2020	

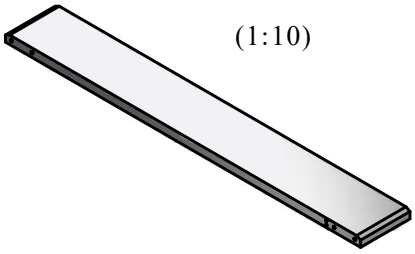




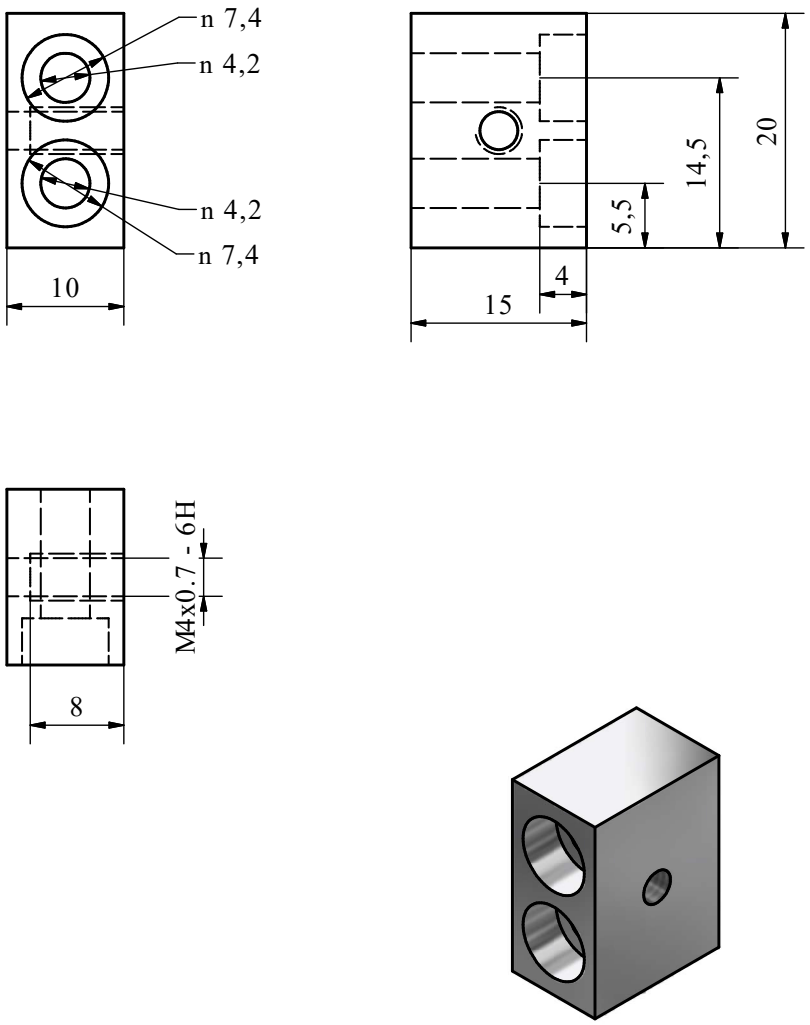


Materiaal: Aluminum		Ruwmaten: Ø 40 x 128	
Norm:		Behandeling:	
Benaming: Rol		Tek. nr: 3	Aantal: 2
		Stuk nr:	Sam. nr:
UHASSELT / KU LEUVEN	Tekenaar: Ruben Berden	Formaat: A4 / Schaal: 1:2	
	Groep:	Datum: 10/05/2020	

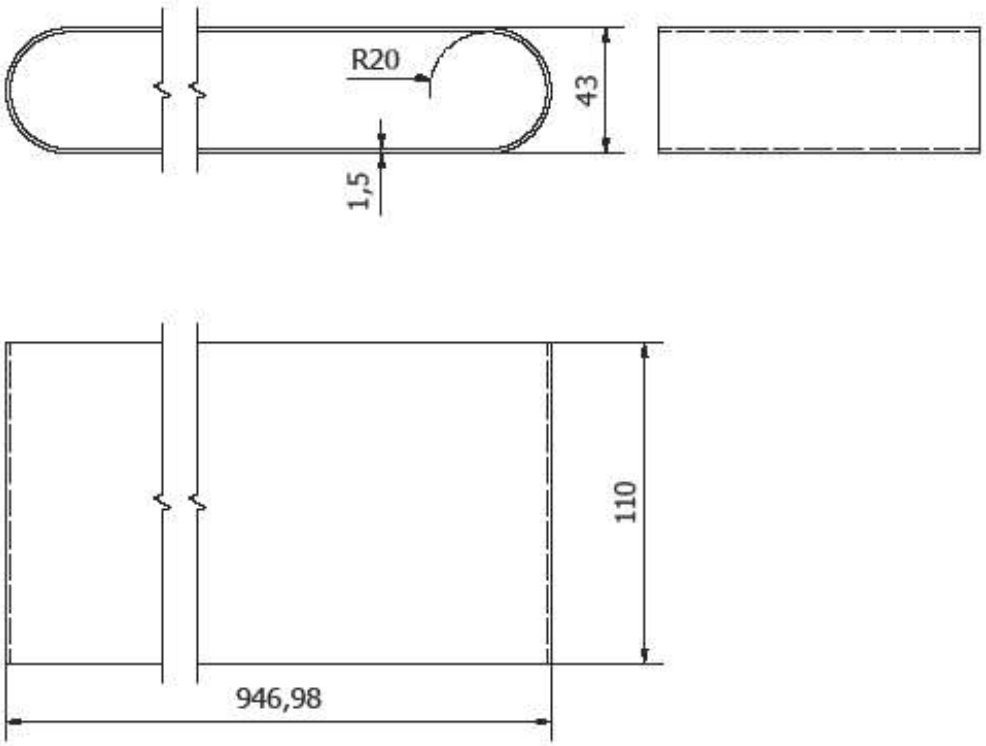




Materiaal: Aluminum		Ruwmaten: 860 x 120 x 20		
Norm:		Behandeling:		
Benaming: Support plate		Tek. nr: 4		Aantal: 1
		Stuk nr:		Sam. nr:
UHASSELT / KU LEUVEN	Tekenaar: Ruben Berden	Formaat: A4	Schaal: 1:3	
	Groep:	Datum: 10/05/2020		



Materiaal: Aluminum		Ruwmaten: 10 x 20 x 15		
Norm:		Behandeling:		
Benaming: Clamping plate		Tek. nr: 5		Aantal: 2
		Stuk nr:		Sam. nr:
UHASSELT / KU LEUVEN	Tekenaar: Ruben Berden	Formaat: A4	Schaal: 2:1	
	Groep:	Datum: 10/05/2020		



Materiaal: Silicon rubber		Ruwmaten:		
Norm:		Behandeling:		
Benaming: Belt		Tek. nr: 6		Aantal: 1
		Stuk nr:		Sam. nr:
UHASSELT	Tekenaar: Ruben Berden	Formaat: A4	Schaal: 1:2	
KU LEUVEN	Groep:	Datum: 10/05/2020		

Attachment B: Technical sheet DC motor

DATI TECNICI
TECHNICAL DATA

serie
series

RH158

Soppressione disturbi con VDR sul collettore
Direzione di rotazione secondo polarità
Può essere montato in ogni posizione
Massimo carico radiale: 50N
Massimo carico assiale: 10N
Temperature di esercizio: -20°C/60°C
Peso approssimativo: 190g

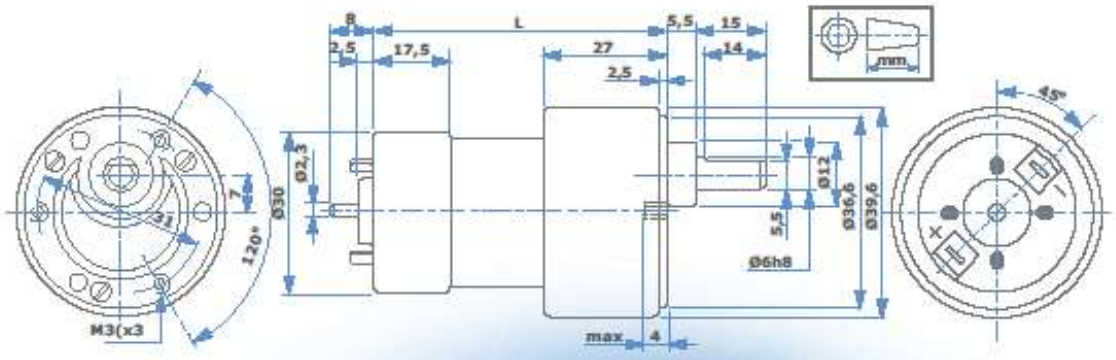
VDR interference suppression on the collector
Direction of rotation depending on polarity
Can be mounted in any position
Maximum radial shaft load: 50N
Maximum axial shaft load: 10N
Temperature range: -20°C/60°C
Approx weight: 190g

Valori tipici a temperatura ambiente +20° Tolleranza +/- 10%

Typical values at ambient temperature +20° Tolerance +/- 10%

TIPO TYPE	TENSIONE NOMINALE NOMINAL VOLTAGE	L	RAPPORTO :1 RATIO TO:1	COPPIA NOMINALE NOMINAL TORQUE	VELOCITÀ SPEED		CURRENT CORRENTE	
					SENZA CARICO NO LOAD	CON COPPIA NOMINALE AT NOMINAL TORQUE	SENZA CARICO NO LOAD	CON COPPIA NOMINALE AT NOMINAL TORQUE
	V	mm		Ncm	rpm		mA	
RH158-12/24-15	12 24	64	14,14	10	440	300	<140 <70	650 330
RH158-12/24-30	12 24	64	29,75	20	210	140	<140 <70	650 330
RH158-12/24-75	12 24	66,5	76,84	50	81	55	<140 <70	680 340
RH158-12/24-100	12 24	66,5	94,37	60	66	45	<140 <70	680 340
RH158-12/24-200	12 24	69	198,5	100	33	23	<140 <70	580 290
RH158-12/24-250	12 24	69	243,8	100	26	21	<140 <70	500 250
RH158-12/24-510	12 24	72	512,85	100	12	10,5	<140 <70	300 150
RH158-12/24-630	12 24	72	629,82	100	10	9	<140 <70	270 135

RH158



RH158

micro motors
Viale Piave, 80/82 - 23879 VERDERIO (LC) ITALY
Tel. 039.510611-499 Fax 039.513617
www.micromotors.eu - info@micromotors.eu