Model Order Reduction on the Differential Algebraic Equations for Conveyor Belt Systems

L. van der Linden

Delft University of Technology

TU Delft

VORTECH

Model Order Reduction on the Differential Algebraic Equations for Conveyor Belt Systems

by

L. van der Linden

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on September 30, 2024 at 13:00.

Student number:	4702697	
Project duration:	December 4, 2023 – September 30, 2024	
Thesis committee:	J. Bentvelsen, MSc.	VORtech, supervisor
	Dr. ir. S. Jain,	TU Delft, supervisor
	Prof. dr. ir. M.B. van Gijzen,	TU Delft, supervisor
	Dr. A.B.T. Barbaro	TU Delft

Cover: An example of a conveyor belt, taken from [1].

TU Delft VORTECH

Preface

After numerous revisions, this report marks the end of my master thesis in Applied Mathematics, and I am grateful to those who have supported and guided me along the way.

First and foremost, I would like to thank my thesis supervisor at VORtech, Joris. Your support and willingness to help me thinking about my dilemmas have been invaluable in keeping me on track throughout this process. I am also thankful to my thesis supervisor, Shobhit, for your persistence in ensuring I thoroughly understood every aspect of my thesis and for sharing your extensive knowledge about model order reduction. To my thesis supervisor Martin, I am grateful for your optimism and constructive criticism. Your helpful feedback and welcoming attitude have greatly contributed to the development of this work. Thank you Shobhit for keeping our meetings with Martin efficient. Thank you Martin for occasionally moving of the subject of the thesis by talking about extraterrestrial creatures, politics and holidays. All the meetings were both enlightening and entertaining.

I also want to thank Alethea Barbaro for joining my thesis committee and taking the time to read this thesis.

Finally, I would like to thank my family and friends for their support over the past nine and a half months. Your encouragement has been crucial in maintaining perspective. A special thanks to my boyfriend for helping me relax and think about life outside of my thesis.

Thank you all for your contributions to this journey. This thesis would not have been possible without your support.

L. van der Linden
Delft, September 2024

Abstract

This thesis explores the improvement of computational efficiency in simulating two-dimensional conveyor belt systems by applying model order reduction (MOR) techniques. Conveyor belts, crucial for material handling in various industries, are traditionally modeled using finite element methods (FEM), which can be computationally demanding, particularly for long-term simulations with a high number of grid elements. To address this, MOR techniques aim to reduce high-dimensional models into lower-dimensional models.

The study investigates three MOR approaches—modal decomposition, Proper Orthogonal Decomposition (POD), and Dynamic Mode Decomposition (DMD)—to apply on existing software built by VORtech that simulates two-dimensional conveyor belt systems. When considering MOR to reduce the two-dimensional system, challenges arise related to nonlinearities, differential algebraic equations (DAEs), and complex modeling steps. Among the methods tested, DMD proved to be the most effective, offering significant reductions in computational time while maintaining accuracy. POD also demonstrated accuracy but had less impact on speed due to the time-consuming complex modeling steps in the simulation software. These complex modeling steps are not investigated in detail in this thesis and therefore not reducible with the intrusive POD. Because of the non-intrusive nature of DMD, this method was able to incorporate these extra processes in the reduced order model.

The study concludes with recommendations for future research, emphasizing the need for optimization of the code segments that handle the complex modeling steps. In addition, conventional modeling approaches as alternatives to the complex interpolation step could be explored, to enhance the applicability of MOR. Furthermore, DMD with control or parametric DMD could be explored to obtain a reduced order model by interpreting misalignments of rollers as controls or parameters. Finally, a method is proposed to make modal decomposition useful for models, where the solutions depend highly on the external forces. Although this method is not applied to the model considered in this thesis, it would be interesting to explore this modified modal decomposition method on other models that are significantly influenced by the external forces.

Nomenclature

The next list describes several symbols that will be used within the body of the document after chapter 2. Most of the symbols are related to the two-dimensional conveyor belt model. Some physical constants will be used in both models considered in this research.

General

$\ddot{x}$	Second order derivative of x with respect to time
$\dot{x}$	First order derivative of x with respect to time
x, z	Coordinate system in the 2D belt space
x^{3D}, y^{3D}, z^{3D}	Coordinate system in the 3D space

Physics values and constants

ϵ_x	Normal strain in x direction
ϵ_z	Normal strain in z direction
γ_{xz}	Shear strain over x, z plane
ν	Poisson's ratio of the belt
Ω	The domain of the belt
ρ	Density of the belt
σ_x	Normal stress in x direction
σ_z	Normal stress in z direction
τ_{xz}	Shear stress over x, z plane
A	Cross-sectional area of the belt
E	Young's modulus of the belt
r	Roller radius
t_b	Thickness of the belt

Variables for roller j

α_{r_j}	Rotation in all directions
$\mathbf{d}_{r_j}$	Translation in all directions
ω_{r_j}	Rotational speed in longitudinal (moving) direction of the belt
$F_{r_j,x}$	Force in x^{3D} direction (similar for y^{3D} and z^{3D} direction)
m_{r_j}	Mass
$T_{r_j,x}$	Torque in x^{3D} direction (similar for y^{3D} and z^{3D} direction)
χ_{r_j}	Vector containing the translations and rotations

Discretization

ℓ_i	Length of element i
$\mathbf{r}_{x,i}$	Radius of roller at node i in the y^{3D}, z^{3D}-plane
$\mathbf{u}$	Displacement of all grid points with alternating direction
Ω_i	The domain of belt segment i
$\partial\Omega_{i,l}$	Left boundary of belt segment i
$\partial\Omega_{i,r}$	Right boundary of belt segment i
m_i	Element mass at node i
n	Total number of unknowns on the grid points
n_η	Number of grid points in z direction
n_ξ	Number of grid points in x direction
n_{f_i}	Number of elements in x direction on free-belt segment i
n_{r_i}	Number of elements in x direction on roller i
u_i	Displacement in x direction of grid point i
v_i	Displacement in z direction of grid point i
w_i	Width of element i
x_i	The x coordinate of grid point i
z_i	The z coordinate of grid point i

Dynamical model

$\bar{\mathbf{u}}_{f_i}$	Displacement of all grid points on free-belt part i (including contactlines)
$\bar{K}_{f_i}$	Stiffness matrix in free-belt segment i including the boundary terms
$\mathbf{u}_{f_i}$	Displacement of all grid points on free-belt part i (excluding contactlines)
$\mathbf{u}_{r_i}$	Displacement of all grid points on roller i (including contactlines)
$\mathbf{x}$	Vector of unknowns that contains: all nodal displacements and roller translations and rotations
F	Forcing vector
F_{f_i}	Forcing vector in free-belt segment i
$F_{x,k}$	Force on node k in x^{3D}-direction (similar for y^{3D} and z^{3D} direction)
K	Stiffness matrix
K_{f_i}	Stiffness matrix in free-belt segment i
M	Mass matrix
M_{f_i}	Mass matrix in free-belt segment i
N	Total number of unknowns

Reduced order models

b	Bandwidth of the stiffness matrix

m	Number of snapshots
N	Dimension of the full order model
N_a	Number of unknowns for the algebraic equations
r	Dimension of the reduced order model

Contents

Acronyms

CFL	Courant-Friedrichs-Lewy
DAE	Differential Algebraic Equation
DMD	Dynamic Mode Decomposition
FEM	Finite Element Methods
MOR	Model Order Reduction
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
POD	Proper Orthogonal Decomposition
RMSE	Root Mean Squared Error
RRMSE	Relative Root Mean Squared Error
ROM	Reduced Order Model
SVD	Singular Value Decomposition

1 Introduction

Conveyor belt systems play a crucial role in the automated movement of materials in various sectors, ranging from manufacturing to logistics [2]. A conveyor belt system is defined as a configuration consisting of rollers that support a single belt. Forces acting on the rollers induce movement, causing the belt to transport materials. Modeling conveyor belt systems is a useful way of predicting movement, forces and stresses on the conveyor belt, without having to construct the physical conveyor belt system. The predicted movement and forces can be used to define constraints for misalignments of rollers before the construction of the physical conveyor belt system. Furthermore, by accurately simulating the conveyor belts through a mathematical model, we can for example minimize downtime [3] or reduce the energy consumption [4].

To determine the dynamic behaviour of a conveyor belt system, finite element methods (FEM) are state-of-the-art [5]. When using FEM to describe the physical behaviour, a model becomes high-dimensional, particularly when using a fine grid discretization to obtain accurate predictions. Therefore, the numerical simulation of the system can be computationally expensive. This is the reason why model order reduction (MOR) methods can be of high usefulness to models that use finite elements.

MOR is a set of techniques that represents the characteristics of a high dimensional problem in a lower dimensional space. The model that describes the dynamics in this lower dimensional space is the reduced order model (ROM). MOR techniques are broadly classified as intrusive or non-intrusive. Intrusive techniques are based on specific knowledge of the governing equations of the dynamical system. Alternatively, the dynamical system can be treated as a black-box and simulation or measurement data may be used to construct reduced order models, obtained via non-intrusive MOR techniques. The choice for which MOR technique to use, depends on specific characteristics of the system. Some common characteristics that are often considered when analyzing a system include the (non-)linearity of the system, the dynamics (properties of the system matrices), the dimension of the system and the dimension of the intended reduced system.

1.1. Related work on conveyor belt systems

The first models on conveyor belt systems focused on the dynamic behaviour in the longitudinal direction, which is the direction of movement of the belt [5]. Because of this, the conveyor belt could be modeled as a one-dimensional string. These models were either analytical models or models where the belt was divided into two elements. With the development of computers, finite element models were assembled where the belt is modeled as a finite number of elements with different masses, spring coefficients and exerted external forces. The interesting cases are mostly the starting and stopping of the movement of the belt (referred to as the transient stages), because here most of the problems occur [6]. The one-dimensional finite element belt model proved effective in analyzing stresses along the longitudinal direction during the starting and stopping phases of the simulation [3]. With this model, an optimal starting phase of a conveyor belt could be described. However, considering the varied loads and resonance within conveyor belt systems, it became imperative to also incorporate modeling in the vertical (or transverse) direction.

Nowadays, FEM models have been developed that simulate conveyor belt movement in longitudinal

and transverse (vertical) direction. The belt is still modeled here as a one-dimensional string. The aim of these simulations can be, for example, to minimize the energy consumption [4] or to prevent excessive belt tensions and resonance of transverse vibration [7].

MOR has been already applied on the dynamics of conveyor belt systems. For example for conveyor belt systems with long belt paths, the number of finite elements is high and therefore it becomes interesting to use MOR. W. Chen et al. [8] have used MOR for a Model Predictive Controller on the linear model of a belt with a length of one kilometer. The balanced truncation method was implemented because the controllability and observability were important characteristics to preserve after reduction. This method was successful in reducing the model order significantly with high accuracy. C. Yang et al. [9] reduced the order of the model simply by using fewer finite elements. This approach was valid because the speed and acceleration could be represented by fewer elements while still maintaining consistency with the original model. G. Suweken, et al [10] have used the modal truncation method on a conveyor belt model that describes the vertical vibrations of a conveyor belt. This modal truncation method did not seem to be able to obtain approximations of the full order model that are valid on long times scales, however.

There are not yet a lot of models that incorporate lateral displacement in conveyor belt systems, next to the longitudinal and transverse displacement. In models that incorporate lateral displacement, the belt cannot be modeled as a one-dimensional string. Instead, it has to be modeled as a two-dimensional plane in a three-dimensional space. G.F.M. Braat's study [11] focused on the contact mechanics of rolling cylinders in conveyor belt systems with two-dimensional belts. Additionally, various control steering systems have been developed for two-dimensional conveyor belt systems [12], [13]. However, these control systems rely on registering the belt's position rather than on a dynamic model capable of predicting movements of the conveyor belt system. Next to this, MOR techniques are only applied on one-dimensional conveyor belt systems. For a two-dimensional belt, the number of finite elements is generally higher than for a one-dimensional belt, because two dimensions have to be discretized instead of one. This makes it even more relevant to explore MOR techniques for models with a two-dimensional belt.

1.2. Contribution and approach

The objective of this research is to speed up existing software built by VORtech that simulates two-dimensional conveyor belt systems by using MOR. This simulation software is developed to refine the design process of a conveyor belt setup by predicting the dynamic behaviour of systems with misaligned rollers. To this end, the conveyor belt system is represented using the finite element method over a two-dimensional belt, capturing dynamics in both the longitudinal and lateral directions [14]. This leads to a large number of degrees of freedom in the finite element model. Moreover, the model explained is designed to simulate the conveyor belt movement to see what happens over a long period, not only during the starting and stopping phase. These two factors contribute to a significant computational load when simulating a single conveyor belt system. Furthermore, since the simulation program is used to investigate the misalignment limitations of rollers, multiple simulations are necessary to cover the full design process of one conveyor belt setup. Consequently, the computational demands are further increased, and the design process for conveyor belt systems with the use of this model is very time-consuming. Therefore, the implementation of MOR techniques is required to accelerate the design process for such conveyor belt systems.

In order to choose the right model order reduction method, it is crucial to understand the characteristics of the system. The equations governing conveyor belt simulations are known as differential algebraic equations (DAEs). These equations combine both differential and algebraic equations into a single system of equations. Unlike simulating partial differential equations (PDEs) or ordinary differential equations (ODEs), simulating DAEs is not as straightforward. Moreover, applying MOR to DAEs requires special treatment of the algebraic equations, which can introduce technical difficulties [15].

Additionally, the system of DAEs contains weak nonlinearities, which can complicate the application of MOR techniques since MOR for nonlinear systems is often more challenging. The POD-Galerkin method, however, is able to handle nonlinear models [16] and DMD can be altered slightly to be applicable to non-linear systems [17].

Lastly, an interpolation step is applied in each time step during the simulation to correct for movements of the grid points. This step provides additional challenges. Model order reduction has been

applied on other models that deal with a moving reference frame [18]. However, the interpolation step is a non-conventional way of modeling and MOR cannot be directly applied on this interpolation step.

To get an understanding of these characteristics and explore how they can be addressed in MOR for the two-dimensional belt model, a simplified one-dimensional belt model is developed. While there are some differences between the one- and two- dimensional model, this one-dimensional model is used to test three different MOR techniques. First, modal decomposition is applied on this one-dimensional model. This method is fully based on the governing equations and is therefore intrusive. Secondly, the Proper Orthogonal Decomposition (POD) is implemented. This method uses simulation data to generate a reduced basis and projects the governing equations on this basis. Because the governing equations are used, this is also an intrusive method. But, because the method also uses simulation data, it is classified as data-driven as well. Finally, a non-intrusive technique, Dynamic Mode Decomposition (DMD), is considered. In this method, simulation data is analysed to compute a set of modes that identify features of the data. These methods are first tested on the one-dimensional model to formulate a conjecture for the applicability on the two-dimensional model. Based on these conjectures, model order reduction will be applied on the two-dimensional model.

1.2.1. Research questions

In this thesis, three MOR techniques for a conveyor belt system will be explored to answer the research question:

To what extent can model order reduction techniques accelerate the existing simulations for a two-dimensional conveyor belt system to enable faster detection of issues arising from manufacturing imperfections?

To answer this question, we have to address the following subquestions.

1. Is it possible to enhance the performance of the existing simulation program for the full order DAEs without yet employing MOR techniques?
2. What is the nature of the nonlinearities in the full order model?
3. What are suitable MOR techniques for accelerating the simulations?
    - In case the systems non-linearities are weak, can we use linear MOR techniques to reduce the two-dimensional conveyor belt systems?
4. How can we handle the interpolation step when applying MOR?

1.3. Outline

This study is structured in the following way. First, we provide background information about the specific conveyor belt systems that are considered in the existing simulation program in chapter 2. The existing simulation program can be used for numerous cases. Therefore the cases of interest are also narrowed down in this chapter. After this, we derive the governing equations for an equivalent one-dimensional conveyor belt setup in chapter 3. In chapter 4, we extend this to the two-dimensional setup that is used in the existing simulation software, but we keep the derivation limited and explain the main difference with the one-dimensional setup. In chapter 5 we discuss the different MOR techniques that will be applied on this system of equations. In chapter 6, we discuss details on the implementation of the methods on the one-dimensional case and show the error results and the speedup. In addition, limitations of methods are discovered and will be explained. In chapter 7, MOR techniques are applied on the two-dimensional model. Results are shown and finally, the methods are compared. In chapter 8, we provide the conclusion of this study and give recommendations for future research.

2

Background on conveyor belt systems

In this chapter the conveyor belt systems of interest are specified. We establish the precision tolerance and explain design choices and terminology in section 2.1. Finally, assumptions used to develop the conveyor belt model are mentioned in section 2.2.

In industrial machinery, multiple conveyor belt systems play crucial roles, as for example is visible in Figure 2.1. These systems primarily serve for transporting materials. The focus for this research is on the conveyor belt system where precision is of most importance. High precision conveyor belts are used in the semiconductor industry, robotics, printing industry and more. Precision alignment of specific belt segments is important, as misalignment can lead to deformed materials or malfunctioning of the machine. These high precision conveyor belts generally have a short belt path. Detailed knowledge of belt displacement is necessary at numerous points, asking for a fine grid discretization used in the finite element method. In the applications for which the simulation program of interest is designed, the accuracy tolerance in both longitudinal and lateral direction is 20 µm.

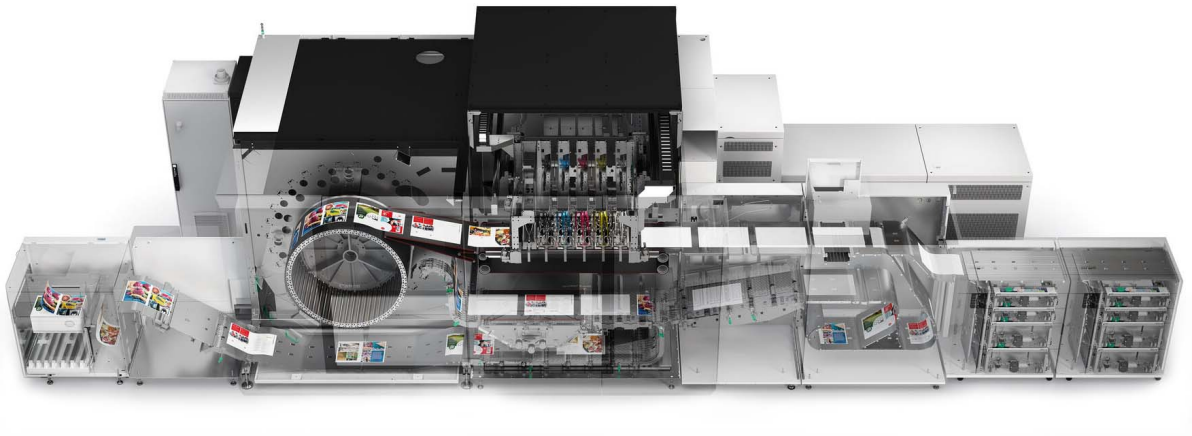
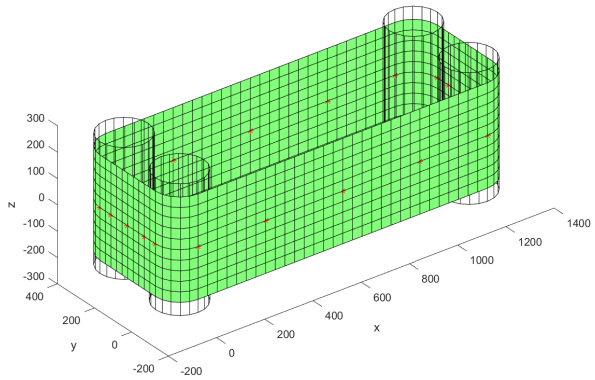


Figure 2.1: The inside of one commercial inkjet printer. Taken from [19].

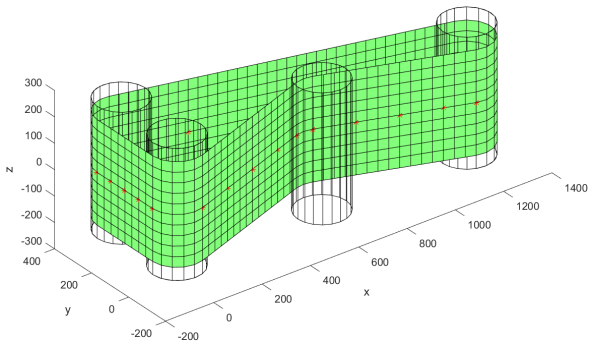
Digital replicas of conveyor belt systems in the simulation program are shown in Figure 2.2. This simulation software is tailored specifically for systems where belt dimensions typically are in the order of meters, contrasting with existing simulation programs designed for belts spanning thousands of meters [8]. Furthermore, while conventional research on conveyor belt systems primarily addresses longitudinal displacement [4], [8], [10], this simulation program incorporates both longitudinal and lateral displacement.

2.1. Design choices and terminology

A conveyor belt system consists of an arrangement of multiple rollers and a singular belt encircling them. Within this setup, one roller serves as the *drive roller*, controlling the rotational velocity for moving the belt around the assembly. The remaining rollers are passive, synchronized with the movement of the belt. These rollers are referred to as *idle rollers*. A system can include both *internal* and *external* rollers, as visually represented in Figure 2.2. A roller positioned within the belt's path is defined as an internal roller, whereas one positioned outside the belt's path is referred to as an external roller.



(a) A system with four internal rollers.



(b) A system with three internal rollers and one external roller located at x=600mm, y=0mm.

Figure 2.2: Two examples of the model of a conveyor-belt system. The values on the axes are given in mm.

To establish a conveyor belt system, the following details are required:

1. *Number of rollers and their characteristics*: Initially, it is necessary to determine the quantity of rollers. For conveyor belts that are considered in this research, the roller count typically ranges from two to six. Parameters such as radius, width, and position for each roller must be specified, along with whether the roller is internal or external. Subsequently, the rollers are positioned according to the orientation depicted in Figure 2.2.

2. *Belt path*: Each roller is assigned a number starting from 1. Once these numbers are assigned, the belt path is established, starting at roller 1. The trajectory of the belt follows the roller numbering, aligning with the internal or external description of each roller.

3. *Belt characteristics*: The belt's width, length, and thickness are determined. The belt width should either match or be smaller than the minimum width of the rollers, because we do not consider cases where a part of the belt cannot be fully attached to the rollers. Next to that, the belt length typically equals the path length, although it is feasible to set a belt length shorter than the path, resulting in initial stress on the belt at the start of the simulation. Additionally, parameters such as Young's modulus (E), Poisson's ratio (ν), density (ρ), and thermal expansivity of the belt material are specified.

For describing the physics of the belt, we make a distinction between *free-belt segments* and *roller segments*. The free-belt segments denote the parts that have no direct contact with the rollers, i.e. the part of the belt that is between two adjacent rollers. Free-belt segments exclude the boundaries, referred to as *contactlines*, where the belt transitions onto or off a roller. The part of the belt that is in direct contact with the rollers is called a roller segment. The roller segments include the associated contactlines.

In reality, a conveyor belt system is never aligned as intended. In the model, this is expressed by tilting or translating the roller by a small amount compared to the supposed setup of the conveyor belt system. The rollers can be rotated around three axes and can be translated over three axes. The axes can be defined as the user wishes, with the constraint that they have to be orthogonal. Without loss of generality, we will define these axes to be in accordance with the standard three unit vectors parallel to the coordinate axes for the rest of this research. When one roller is in standard position, without any tilt or translation, the three axes are oriented as shown in Figure 2.3. These axes are oriented in the same way as in Figure 2.2. Using these three axes, a local coordinate system is created per roller.

The origin of the coordinate system is the center of the roller. For each of the three directions, the roller rotates around the specific axis that goes through the center of the roller. By combining translations and rotations, the roller can reach all positions and orientations in the 3D space.

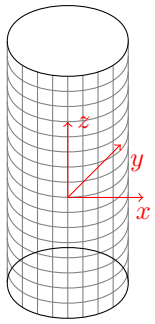


Figure 2.3: Axes of rotation and translation of one roller.

A **translation** of a roller is defined by setting an *initial translation* together with a *translational velocity*. The initial translation gives the roller a different position than the intended position (defined during the setup of the belt system). During a dynamic simulation, the translation is altered with the translational velocity. If the translational velocity is zero, the roller has a static translation. A static translation is the only translation of interest in our case: when a translational velocity is set, the roller will eventually move from the intended position substantially. Because this research is intended to simulate long dynamical simulations with small misalignments, this is not a case of interest.

A **rotation** is defined by setting an *initial rotation* together with a *rotational velocity*. The initial rotation gives the roller a small tilted orientation. During a dynamic simulation, the rotation is altered with the rotational velocity. If the rotational velocity is zero, the roller has a static rotation. Around the z direction, it is common to set a rotational velocity: this makes the belt move around the rollers. In the x and y direction, however, only a small tilt from the original upright position is possible. So the x and y rotations have to be bounded. Therefore it is unlikely that a rotational velocity has to be set for these directions. For the rest of this research, we will assume there is no rotational velocity in the x and y direction. The static rotations are the most relevant in this analysis, as they define a misalignment in the setup of the conveyor belt system, so this assumption does not limit this research.

2.2. Assumptions

To model conveyor belt systems, some assumptions are made. First of all, the impact of loads on the belt are neglected. This simplification is justified by the negligible mass of transported materials in the systems of interest in this research, compared to the other masses in the systems. Consequently, we do not simulate transverse belt displacement in our model, although it may be relevant for systems with long belts [10]. Additionally, we assume that there is no slip between the belt and the rollers, allowing us to directly relate the movement of a point on the belt with the corresponding movement of the roller at that point. This assumption is reasonable as the rollers are made from materials ensuring minimal relative motion between the belt and the roller. Lastly, we assume the absence of gravity in the system.

3

One-dimensional model of a conveyor belt system

To develop an understanding of the model that simulates belt movement, we derive the dynamic equations for a simple one-dimensional belt system. First the finite element method is explained for the equations governing the motion on the free-belt segments in section 3.1. Next, we consider the algebraic constraints that define the boundary conditions in section 3.2. In section 3.3, the system of equations is assembled and in section 3.4, different approaches are outlined to solve this system of equations. Finally, in section 3.5, the interpolation step for this one-dimensional model is explained.

In this one-dimensional model, the belt is modeled as a string with a cross-sectional area A. This string has the same length as the length of the belt path, that is the path over which the string moves around the roller. The two rollers have the same radius r and roller 1 is the drive roller that rotates with a constant velocity of ω_{r_1} radians per second. The other roller is an idle roller, which means that it moves along with the belt without slipping and that there is no torque applied on this body, i.e. we have $T_{r_2} = 0$ on the right roller in Figure 3.1. The displacement of the belt is of interest and will be determined with finite element analysis. In the finite element equations, nodal displacements are the unknowns. In Figure 3.1 the node indexing is shown. The index starts at the incoming contactline of roller 1 and increases with the direction of movement of the belt. n is the total number of elements, n_{r_i} is the number of elements on roller i and n_{f_i} is the number of elements on free-belt segment i.

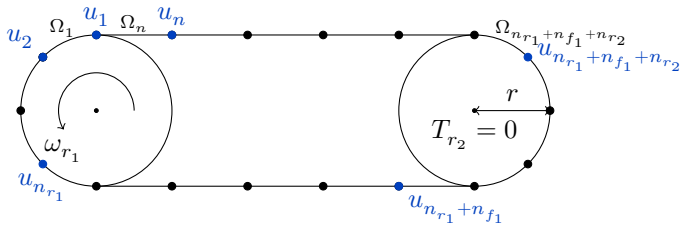


Figure 3.1: Schematic drawing of a one dimensional setup of a belt moving around two rollers with the same radius. The left roller is roller 1, the right roller is roller 2.

The movement of the belt is modeled with a dynamic equations on the free-belt segments and an algebraic description of the movement on the roller segments. The dynamic equations are a result of finite element analysis on a PDE. The algebraic equations and the dynamical equations are combined in the full system of equations, a differential algebraic equation.

3.1. Free-belt segments

Newton's second law results in the equation of motion on the free-belt segments [20],

$$\rho \frac{d^2\mathbf{u}}{dt^2} - \nabla \cdot \sigma = \mathbf{f}. \tag{3.1}$$

where $\mathbf{u}$ is the displacement vector that depends on the position vector $\mathbf{x}$ and time t, σ is the stress tensor and $\mathbf{f}$ is the body force per unit volume. That is, $\mathbf{f} = \frac{f}{A}$, where f is the body force per unit length. In one dimension, we have

$$\rho \frac{d^2u}{dt^2} - \frac{d\sigma_x}{dx} = \frac{f}{A}, \tag{3.2}$$

where σ_x is the strain in x direction. From Hooke's law, the strain/stress equation in one dimension is

$$\sigma_x = E\epsilon_x, \tag{3.3}$$

where E is the Young's coefficient of the belt material and ϵ_x is the strain over the x direction given by

$$\epsilon_x = \frac{du}{dx}. \tag{3.4}$$

This gives the PDE

$$\rho A \frac{d^2u}{dt^2} - EA\frac{d^2u}{dx^2} = f. \tag{3.5}$$

In this finite element analysis, Equation 3.5 is referred to as the strong form. To derive the finite element equations, we first have to derive the weak form. For this, we enforce that the weighted residual has to be zero in a pointwise manner, that is

$$\int_\Omega \rho A \frac{d^2u}{dt^2} w \, dx - \int_\Omega EA\frac{d^2u}{dx^2} w \, dx - \int_\Omega fw \, dx = 0, \tag{3.6}$$

where w are the test functions that have to be in some test space. We use the constraint $w|_{\partial\Omega} = 0$ on this test space to derive

$$\int_\Omega \rho A \frac{d^2u}{dt^2} w \, dx + \int_\Omega EA\frac{du}{dx}\frac{dw}{dx} \, dx - \int_\Omega fw \, dx = 0, \tag{3.7}$$

with integration by parts on the second integral. This is the continuous weak form. The solution to this continuous problem will be approximated using finite elements.

3.1.1. Meshing the domain

For setting up the system of finite element equations, the free-belt segment is partitioned into a number of elements, as is visible in Figure 3.1. One mesh element is depicted in Figure 3.2 and is defined to be the open interval (x_i, x_{i+1}). With this mesh, the finite element solution is approximated. The following

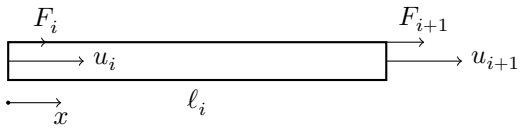


Figure 3.2: Element i in the one dimensional belt of length ℓ_i, with cross-sectional area A and density ρ. The external forces are given separately on both nodes. u_i is the nodal displacement.

analysis will be done for the first free-belt segment, but is similar for the second free-belt segment.

3.1.2. Construction of finite dimensional function spaces

Now that we have created a mesh for Ω, we will use it to define finite dimensional function spaces on Ω. One of the most common choices for a finite element space is used, namely the *hat functions*. That is, for each node we have one linear basis function that is 1 on the node itself and 0 on the rest of the nodes:

$$N_i(x) = \begin{cases} 1 - \frac{1}{\ell_{i-1}}(x_i - x) & \text{if } x_{i-1} < x < x_i \\ 1 - \frac{1}{\ell_i}(x - x_i) & \text{if } x_i < x < x_{i+1} \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

The first and last basis function on one free-belt element is half a hat function, that is, for example for the first basis function on the first free-belt segment:

$$N_{n_{r_1}+1}(x) = \begin{cases} 1 - \frac{1}{\ell_{n_{r_1}+1}}(x - x_{n_{r_1}+1}) & \text{if } x_{n_{r_1}+1} < x < x_{n_{r_1}+2}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

Any function, in particular the solution and test functions u and w, can be approximated by a linear combination of these basis functions. The function u_h is the discretized solution of the weak form. For the first free-belt segment, we have

$$u_h(x,t) = \sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}+1} u_i(t) N_i(x), \tag{3.10}$$

$$w_h(x,t) = \sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}+1} w_i(t) N_i(x). \tag{3.11}$$

When we have derived the weak form, we have used the constraint $w|_{\partial\Omega} = 0$. This means that the factors $w_{n_{r_1}+1}(t)$ and $w_{n_{r_1}+n_{f_1}+1}(t)$ are zero. Thus, we have for w_h:

$$w_h(x,t) = \sum_{i=n_{r_1}+2}^{n_{r_1}+n_{f_1}} w_i(t) N_i(x). \tag{3.12}$$

The boundary conditions for one free-belt segment are expressed as algebraic equations. These conditions are assumed to be known for the purpose of constructing the finite element equations. So the functions $u_{n_{r_1}+1}(t)$ and $u_{n_{r_1}+n_{f_1}+1}(t)$ for the first free-belt segment are provided as time-dependent values. Let us assume with

$$u_{n_{r_1}+1}(t) = g_0(t), \tag{3.13}$$

$$u_{n_{r_1}+n_{f_1}+1}(t) = g_L(t). \tag{3.14}$$

This results in the discretized solution of the form

$$u_h(x,t) = g_0(t) N_{n_{r_1}+1}(x) + \sum_{i=n_{r_1}+2}^{n_{r_1}+n_{f_1}} u_i(t) N_i(x) + g_L(t) N_{n_{r_1}+n_{f_1}+1}(x). \tag{3.15}$$

3.1.3. Discretized weak problem

When we substitute these in the continuous weak problem in Equation 3.7, we derive the discretized weak problem. The solution has to satisfy the weak problem for every function w_h in the test space. Because the (inner) hat functions are a basis for this test space, we can say that the weak problem will be satisfied for all functions, when it is satisfied for all the basis functions with

$$\int_\Omega \rho A \frac{d^2}{dt^2} \left(\sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}+1} u_i(t) N_i(x) \right) N_j(x)\, dx + \int_\Omega EA \frac{d}{dx} \left(\sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}+1} u_i(t) N_i(x) \right) \frac{dN_j}{dx}(x)\, dx \tag{3.16}$$

$$- \int_\Omega f \cdot N_j(x) = 0 \quad \forall j = n_{r_1}+2, ..., n_{r_1}+n_{f_1}.$$

By interchanging the integral and the sum and taking the variables independent of x out of the integral, we get

$$\sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}+1} \rho A \frac{d^2u_i}{dt^2}(t) \int_\Omega N_i(x)N_j(x)\,dx + \sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}+1} EAu_i(t) \int_\Omega \frac{dN_i}{dx}(x)\frac{dN_j}{dx}(x)\,dx - \int_\Omega fN_j(x)\,dx = 0 \quad \forall j = n_{r_1}+2, ..., n_{r_1}+n_{f_1}. \tag{3.17}$$

The terms where all values are known, so u_i on the boundaries and f, will be put in the right hand side. Then the equations can be written into a linear system for the unknowns u_i and $\frac{d^2u_i}{dt^2} = \ddot{u}_i$, given by

$$\underbrace{\begin{bmatrix} \rho A\int_\Omega N_{n_{r_1}+2}(x)N_{n_{r_1}+2}(x)dx & \dots & \rho A\int_\Omega N_{n_{r_1}+n_{f_1}}(x)N_{n_{r_1}+2}(x)dx \\ \vdots & \ddots & \vdots \\ \rho A\int_\Omega N_{n_{r_1}+2}(x)N_{n_{r_1}+n_{f_1}}(x)dx & \dots & \rho A\int_\Omega N_{n_{r_1}+n_{f_1}}(x)N_{n_{r_1}+n_{f_1}}(x)dx \end{bmatrix}}_{M_{f_1}} \begin{bmatrix} \ddot{u}_{n_{r_1}+2}(t) \\ \ddot{u}_{n_{r_1}+3}(t) \\ \vdots \\ \ddot{u}_{n_{r_1}+n_{f_1}}(t) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} EA\int_\Omega \frac{dN_{n_{r_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+2}}{dx}(x)\,dx & \dots & EA\int_\Omega \frac{dN_{n_{r_1}+n_{f_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+2}}{dx}(x)\,dx \\ \vdots & \ddots & \vdots \\ EA\int_\Omega \frac{dN_{n_{r_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\,dx & \dots & EA\int_\Omega \frac{dN_{n_{r_1}+n_{f_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\,dx \end{bmatrix}}_{K_{f_1}} \begin{bmatrix} u_{n_{r_1}+2}(t) \\ u_{n_{r_1}+3}(t) \\ \vdots \\ u_{n_{r_1}+n_{f_1}}(t) \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} \int_\Omega fN_{n_{r_1}+2}(x)\,dx \\ \vdots \\ \int_\Omega fN_{n_{r_1}+n_{f_1}}(x)\,dx \end{bmatrix}}_{F_{f_1}}$$

$$- \underbrace{\begin{bmatrix} \rho A\int_\Omega N_{n_{r_1}+1}(x)N_{n_{r_1}+2}(x)dx & \rho A\int_\Omega N_{n_{r_1}+n_{f_1}+1}(x)N_{n_{r_1}+2}(x)dx \\ \vdots & \vdots \\ \rho A\int_\Omega N_{n_{r_1}+1}(x)N_{n_{r_1}+n_{f_1}}(x)dx & \rho A\int_\Omega N_{n_{r_1}+n_{f_1}+1}(x)N_{n_{r_1}+n_{f_1}}(x)dx \end{bmatrix}}_{G_{M,f_1}} \begin{bmatrix} \ddot{g}_0(t) \\ \ddot{g}_L(t) \end{bmatrix}$$

$$- \underbrace{\begin{bmatrix} EA\int_\Omega \frac{dN_{n_{r_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+2}}{dx}(x)\,dx & EA\int_\Omega \frac{dN_{n_{r_1}+n_{f_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+2}}{dx}(x)\,dx \\ \vdots & \vdots \\ EA\int_\Omega \frac{dN_{n_{r_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\,dx & EA\int_\Omega \frac{dN_{n_{r_1}+n_{f_1}+1}}{dx}(x)\frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\,dx \end{bmatrix}}_{G_{K,f_1}} \begin{bmatrix} g_0(t) \\ g_L(t) \end{bmatrix}. \tag{3.18}$$

3.1.4. Assembly of equation matrices

The integrals in Equation 3.18 will be evaluated using local assembly. The integral over Ω is broken up into integrals over the elements Ω_i. These i-th element integrals are nonzero on a few elements, since every $N_i(x)$ and $\frac{dN_i}{dx}$ is only nonzero for $x \in \Omega_{i-1} \cup \Omega_i$. In the end, the total mass matrix M_{f_i} is the sum of the local mass matrices,

$$M_{f_1} = \sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}} M_{f_1}|_{\Omega_i}, \tag{3.19}$$

where

$$M_{f_1}|_{\Omega_i} = \begin{bmatrix} \rho A\int_{\Omega_i} N_{n_{r_1}+2}(x)N_{n_{r_1}+2}(x)dx & \dots & \rho A\int_{\Omega_i} N_{n_{r_1}+n_{f_1}}(x)N_{n_{r_1}+2}(x)dx \\ \vdots & \ddots & \vdots \\ \rho A\int_{\Omega_i} N_{n_{r_1}+2}(x)N_{n_{r_1}+n_{f_1}}(x)dx & \dots & \rho A\int_{\Omega_i} N_{n_{r_1}+n_{f_1}}(x)N_{n_{r_1}+n_{f_1}}(x)dx \end{bmatrix}. \tag{3.20}$$

Similarly for the stiffness matrix K_{f_1}, we have

$$K_{f_1} = \sum_{i=n_{r_1}+1}^{n_{r_1}+n_{f_1}} K_{f_1}|_{\Omega_i}, \tag{3.21}$$

$$K_{f_1}|_{\Omega_i} = \begin{bmatrix} EA\int_{\Omega_i} \frac{dN_{n_{r_1}+2}}{dx}(x)\frac{dN_{n_{r_1}+2}}{dx}(x)\,dx & \dots & EA\int_{\Omega_i} \frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\frac{dN_{n_{r_1}+2}}{dx}(x)\,dx \\ \vdots & \ddots & \vdots \\ EA\int_{\Omega_i} \frac{dN_{n_{r_1}+2}}{dx}(x)\frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\,dx & \dots & EA\int_{\Omega_i} \frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\frac{dN_{n_{r_1}+n_{f_1}}}{dx}(x)\,dx \end{bmatrix}. \tag{3.22}$$

When we look at any element on one free-belt part Ω_m, only N_m and N_{m+1} are nonzero on this element. Therefore $M_{f_1}[i,j]|_{\Omega_m} = 0$ and $K_{f_1}[i,j]|_{\Omega_m} = 0$ when i and j are not m or $m+1$ [1]. For the other terms in $M_{f_1}|_{\Omega_m}$, we have

$$M_{f_1}[m,m+1]|_{\Omega_m} = M_{f_1}[m+1,m]|_{\Omega_m} = \rho A\int_{\Omega_m} N_m(x)N_{m+1}(x)dx \approx 0. \tag{3.23}$$

by the Newton-Cotes rule which says

$$\int_{\Omega_m} N_m(x)N_{m+1}(x)dx \approx \frac{1}{2}\ell_m\left[N_m(x_m)N_{m+1}(x_m) + N_m(x_{m+1})N_{m+1}(x_{m+1})\right] = 0, \tag{3.24}$$

since $N_{m+1}(x_m) = 0$ and $N_m(x_{m+1}) = 0$. Next to that, we have

$$M_{f_1}[m,m]|_{\Omega_m} = \int_{\Omega_m} N_m^2(x)dx \approx \frac{1}{2}\rho A\ell_m, \tag{3.25}$$

$$M_{f_1}[m+1,m+1]|_{\Omega_m} = \int_{\Omega_m} N_{m+1}^2(x)dx \approx \frac{1}{2}\rho A\ell_m. \tag{3.26}$$

by Newton-Cotes and the fact that $N_m(x_m) = N_{m+1}(x_{m+1}) = 1$ and $N_m(x_{m+1}) = N_{m+1}(x_m) = 0$. This is the same result when taking the lumped mass as in [21]. For the terms in $K_{f_1}|_{\Omega_m}$, we have

$$K_{f_1}[m,m+1]|_{\Omega_m} = K_{f_1}[m+1,m]|_{\Omega_m} = EA\int_{\Omega_m} \frac{dN_m}{dx}\frac{dN_{m+1}}{dx} = EA\int_{\Omega_m} -\frac{1}{\ell_m}\cdot\frac{1}{\ell_m}dx \tag{3.27}$$

$$= -\frac{EA}{(\ell_m)^2}(x_{m+1}-x_m) = -\frac{EA}{\ell_m}. \tag{3.28}$$

Next to that, we have

$$K_{f_1}[m,m]|_{\Omega_m} = EA\int_{\Omega_m} (\frac{dN_m}{dx})^2dx = EA\int_{\Omega_m}\left(-\frac{1}{\ell_m}\right)^2 dx = \frac{EA}{(\ell_m)^2}(x_{m+1}-x_m) = \frac{EA}{\ell_m}, \tag{3.29}$$

$$K_{f_1}[m+1,m+1]|_{\Omega_m} = EA\int_{\Omega_m} (\frac{dN_{m+1}}{dx})^2dx = EA\int_{\Omega_m}\left(\frac{1}{\ell_m}\right)^2 dx = \frac{EA}{(\ell_m)^2}(x_{m+1}-x_m) = \frac{EA}{\ell_m}. \tag{3.30}$$

And likewise, we get

$$G_{K,f_1} = \begin{bmatrix} -\frac{1}{\ell_{n_{r_1}+1}} & 0 \\ \vdots & \vdots \\ 0 & -\frac{1}{\ell_{n_{r_1}+n_{f_1}}} \end{bmatrix}, \tag{3.31}$$

$$G_{M,f_1} = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}. \tag{3.32}$$

[1] Note that we simplified the indices of matrix M_{f_1} and K_{f_1}: by index i,j, we actually mean index $i-n_{r_1}-1, j-n_{r_1}-1$. This simplification will be used in the rest of this subsection.

3.1.5. Assembly of system

The value f is the body force per unit length. On the free-belt segments in this one-dimensional model, this is given to be zero. Therefore, the right-hand side only consists of the boundary values. The element stiffness matrices and the element mass matrices are combined to the general stiffness and mass matrix on the free-belt segments. For the first free-belt segment, we call the resulting matrices K_{f_1} and M_{f_1}.

$$K_{f_1} = EA \begin{bmatrix} \frac{1}{\ell_{n_{r_1}+1}} + \frac{1}{\ell_{n_{r_1}+2}} & -\frac{1}{\ell_{n_{r_1}+2}} & 0 & \dots \\ -\frac{1}{\ell_{n_{r_1}+2}} & \frac{1}{\ell_{n_{r_1}+2}} + \frac{1}{\ell_{n_{r_1}+3}} & -\frac{1}{\ell_{n_{r_1}+3}} & \\ & -\frac{1}{\ell_{n_{r_1}+3}} & \frac{1}{\ell_{n_{r_1}+3}} + \frac{1}{\ell_{n_{r_1}+4}} & -\frac{1}{\ell_{n_{r_1}+4}} \\ & \vdots & \ddots & \ddots \\ & & -\frac{1}{\ell_{n_{r_1}+n_{f_1}-1}} & \frac{1}{\ell_{n_{r_1}+n_{f_1}-1}} + \frac{1}{\ell_{n_{r_1}+n_{f_1}}} \end{bmatrix}.$$

$$M_{f_1} = \frac{1}{2}\rho A \begin{bmatrix} \ell_{n_{r_1}+1} + \ell_{n_{r_1}+2} & 0 & 0 & \dots & 0 \\ 0 & \ell_{n_{r_1}+2} + \ell_{n_{r_1}+3} & 0 & & \vdots \\ & 0 & \ell_{n_{r_1}+3} + \ell_{n_{r_1}+4} & \dots & \\ & \vdots & \ddots & \ddots & \\ & & & \ell_{n_{r_1}+n_{f_1}} + \ell_{n_{r_1}+n_{f_1}+1} & \end{bmatrix}.$$

The dynamics on one free-belt segment can now be described as

$$K_f \mathbf{u}_f + M_f \ddot{\mathbf{u}}_f = -G_{K,f} \begin{bmatrix} g_0(t) \\ g_L(t) \end{bmatrix}. \tag{3.33}$$

3.2. Roller segments

In the dynamical systems in Equation 3.33, we still have to define the boundary conditions. The boundary conditions describe the displacement of the two nodes on the adjacent contactlines. The displacement of the grid points of the roller is a result of the rotation of the roller with the relation,

$$\dot{u}_r(t) = r\dot{\alpha}_r(t). \tag{3.34}$$

The given boundary condition does not correspond to a Neumann boundary condition. Neumann conditions typically involve derivatives with respect to spatial coordinates, whereas the provided condition involves a derivative with respect to time. Consequently, it cannot be directly incorporated into the system of equations. To address this, we reinterpret it to have a modified Dirichlet boundary condition. We reformulate the condition by discretizing in time using Backward Euler:

$$\begin{aligned} u_{r_i}(t) &= u_{r_i}(t-\Delta t) + \Delta t \dot{u}_{r_i}(t) \\ u_{r_i}(t) &= u_{r_i}(t-\Delta t) + \Delta t r \dot{\alpha}_{r_i}(t) \\ u_{r_i}(t) &= u_{r_i}(t-\Delta t) + \Delta t r \left(\frac{\alpha_{r_i}(t) - \alpha_{r_i}(t-\Delta t)}{\Delta t} \right) \\ u_{r_i}(t) - r\alpha_{r_i}(t) &= u_{r_i}(t-\Delta t) - r\alpha_{r_i}(t-\Delta t). \end{aligned} \tag{3.35}$$

In the configuration treated in this chapter, the rotation of the first roller is given and and can be substituted here directly in α_{r_1}. However, the rotation of the second roller is unknown and is determined by the torque/force balance on the roller. Without loss of generality, we assume that the roller rotation is unknown for all rollers. Therefore, in Equation 3.33, $g_0(t)$ and $g_L(t)$ are not given functions in time, but dependent on the unknowns. Hence, we will not include this boundary condition in the right-hand side. In the existing simulation program, the circumvention of this issue is including u_r and α in the set

of unknowns and therefore $G_{K,f}$ and K_{f_1} will be merged into

$$\bar{K}_{f_1} = EA \begin{bmatrix} -\frac{1}{\ell_{n_{r_1}+1}} & \frac{1}{\ell_{n_{r_1}+1}} + \frac{1}{\ell_{n_{r_1}+2}} & -\frac{1}{\ell_{n_{r_1}+2}} & 0 & \dots & 0 \\ & -\frac{1}{\ell_{n_{r_1}+2}} & \frac{1}{\ell_{n_{r_1}+2}} + \frac{1}{\ell_{n_{r_1}+3}} & -\frac{1}{\ell_{n_{r_1}+3}} & & \vdots \\ & & -\frac{1}{\ell_{n_{r_1}+3}} & \frac{1}{\ell_{n_{r_1}+3}} + \frac{1}{\ell_{n_{r_1}+4}} & -\frac{1}{\ell_{n_{r_1}+4}} & \\ & & \vdots & \ddots & \ddots & \\ & & & -\frac{1}{\ell_{n_{r_1}+n_{f_1}-1}} & \frac{1}{\ell_{n_{r_1}+n_{f_1}-1}} + \frac{1}{\ell_{n_{r_1}+n_{f_1}}} & -\frac{1}{\ell_{n_{r_1}+n_{f_1}}} \end{bmatrix}.$$

This results in the dynamical equation

$$\bar{K}_f \bar{\mathbf{u}}_f + M_f \ddot{\bar{\mathbf{u}}}_f = 0. \tag{3.36}$$

Note that we use $\bar{\mathbf{u}}_f$ here, because the system is also dependent on the displacement of the two nodes on the adjacent contactlines. The relation between the unknowns u_r and α is added in the full set of equations as an algebraic equation.

3.2.1. Torque balance

We assume that there is no slip between the rollers and the belt. This is modeled by pinning the belt to the rollers, that is, the belt movement is directly related to the movement of the roller. In this case, the stress tensor does not contribute to the forces. Therefore, we only have a contribution of the acceleration and mass in the force on the belt. The total torque/force on one roller can either be specified so that the total forces have to be exactly equal to this value, or it can be a result of the displacements.

On the idle roller (roller 2), an external applied torque is given. The total torque is the sum of the external applied torque and the torque applied by the belt. The torque applied on the belt is a result of the force applied on the belt and the radius of the roller,

$$T = \underbrace{T_{external}}_{=0} + \underbrace{T_{belt}}_{=-r\times F}. \tag{3.37}$$

The total torque has to be $I\ddot{\alpha}$, where $I = \frac{1}{2} m_r r^2$, assuming the roller is a solid cylinder with radius r and mass m_r. This results in the equation

$$r \times F + \frac{1}{2} m_r r^2 \ddot{\alpha} = \underbrace{T_{external}}_{=0}. \tag{3.38}$$

F is the sum of the local nodal forces, $F_j = \int_\Omega f N_j(x)\, dx$. The nodal forces are calculated by the acceleration and the lumped mass of the nodes on the roller. The nodal force on the contactline also consists of a stiffness term. For roller 2, this is

$$\begin{bmatrix} F_{n_{r_1}+n_{f_1}+1} \\ F_{n_{r_1}+n_{f_1}+2} \\ \vdots \\ F_{n_{r_1}+n_{f_1}+n_{r_2}+1} \end{bmatrix} = \begin{bmatrix} -\frac{AE}{\ell_{n_{r_1}+n_{f_1}}} & \frac{AE}{\ell_{n_{r_1}+n_{f_1}}} & 0 & \dots & 0 \\ 0 & 0 & & & \vdots \\ \vdots & & \ddots & & \\ 0 & \dots & 0 & \frac{AE}{\ell_{n_{r_1}+n_{f_1}+n_{r_2}+1}} & -\frac{AE}{\ell_{n_{r_1}+n_{f_1}+n_{r_2}+1}} \end{bmatrix} \begin{bmatrix} u_{n_{r_1}+n_{f_1}} \\ u_{n_{r_1}+n_{f_1}+1} \\ u_{n_{r_1}+n_{f_1}+2} \\ \vdots \\ u_{n_{r_1}+n_{f_1}+n_{r_2}+1} \\ u_{n_{r_1}+n_{f_1}+n_{r_2}+2} \end{bmatrix}$$

$$+ \frac{\rho A}{2} \begin{bmatrix} \ell_{n_{r_1}+n_{f_1}} + \ell_{n_{r_1}+n_{f_1}+1} & 0 & 0 \\ 0 & \ell_{n_{r_1}+n_{f_1}+1} + \ell_{n_{r_1}+n_{f_1}+2} & \\ \ddots & \ddots & \ddots \\ & 0 & \ell_{n_{r_1}+n_{f_1}+n_{r_2}} + \ell_{n_{r_1}+n_{f_1}+n_{r_2}+1} \end{bmatrix} \begin{bmatrix} \ddot{u}_{n_{r_1}+n_{f_1}+1} \\ \ddot{u}_{n_{r_1}+n_{f_1}+2} \\ \vdots \\ \ddot{u}_{n_{r_1}+n_{f_1}+n_{r_2}+1} \end{bmatrix}.$$

The torque applied by the belt should be equal to the column sum of the system above multiplied with

the radius r, that is,

$$T_{belt} = -r \times F$$

$$= \underbrace{-r \begin{bmatrix} -\frac{AE}{\ell_{n_{r_1}+n_{f_1}}} & \frac{AE}{\ell_{n_{r_1}+n_{f_1}}} & 0 & \dots & 0 & \frac{AE}{\ell_{n_{r_1}+n_{f_1}+n_{r_2}+1}} & -\frac{AE}{\ell_{n_{r_1}+n_{f_1}+n_{r_2}+1}} \end{bmatrix}}_{=:K_{T_{r_2}}} \begin{bmatrix} u_{n_{r_1}+n_{f_1}} \\ u_{n_{r_1}+n_{f_1}+1} \\ u_{n_{r_1}+n_{f_1}+2} \\ \vdots \\ u_{n_{r_1}+n_{f_1}+n_{r_2}+1} \\ u_{n_{r_1}+n_{f_1}+n_{r_2}+2} \end{bmatrix} \tag{3.39}$$

$$- \underbrace{r\frac{\rho A}{2} \begin{bmatrix} (\ell_{n_{r_1}+n_{f_1}} + \ell_{n_{r_1}+n_{f_1}+1}) & \dots & (\ell_{n_{r_1}+n_{f_1}+n_{r_2}} + \ell_{n_{r_1}+n_{f_1}+n_{r_2}+1}) \end{bmatrix}}_{=:M_{T_{r_2}}} \begin{bmatrix} \ddot{u}_{n_{r_1}+n_{f_1}+1} \\ \ddot{u}_{n_{r_1}+n_{f_1}+2} \\ \vdots \\ \ddot{u}_{n_{r_1}+n_{f_1}+n_{r_2}+1} \end{bmatrix}.$$

3.3. System of equations

Every displacement of the belt on the roller is treated as unknowns in the system of equations. Next to that, all the roller rotations are treated as unknowns in the system. The full system of equations consists of the equations for the dynamics on the free-belt segments (Equation 3.36), equations for the displacement on the roller segments (Equation 3.35) and the equations that determine the rotation (indirectly with Equation 3.38 or trivially when it is given with $\alpha_{r_i}(t) = t \cdot \omega_{r_i}$). For the system in Figure 3.1, where α_{r_1} and T_{r_2} are given, this is given as

$$\begin{bmatrix} I & & & & -r\mathbf{1} & \\ & \bar{K}_{f_1} & & & & \\ & & I & & & -r\mathbf{1} \\ & & & \bar{K}_{f_2} & & \\ & & & & 1 & \\ & & K_{T_{r_2}} & & & \end{bmatrix} \begin{bmatrix} \mathbf{u}_{r_1} \\ \mathbf{u}_{f_1} \\ \mathbf{u}_{r_2} \\ \mathbf{u}_{f_2} \\ \alpha_{r_1} \\ \alpha_{r_2} \end{bmatrix} \tag{3.40}$$

$$+ \begin{bmatrix} & M_{f_1} & & & & \\ & & & & & \\ & & & M_{f_2} & & \\ & & & & & \\ & & M_{T_{r_2}} & & \frac{1}{2}m_r r^2 & \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_{r_1} \\ \ddot{\mathbf{u}}_{f_1} \\ \ddot{\mathbf{u}}_{r_2} \\ \ddot{\mathbf{u}}_{f_2} \\ \ddot{\alpha}_{r_1} \\ \ddot{\alpha}_{r_2} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{r_1}(t-\Delta t) - r\alpha_{r_1}(t-\Delta t) \\ 0 \\ \mathbf{u}_{r_2}(t-\Delta t) - r\alpha_{r_2}(t-\Delta t) \\ 0 \\ t\omega_{r_1} \\ 0 \end{bmatrix},$$

where $\mathbf{1}$ is a column-vector with ones. Since we combine algebraic equations with dynamical equations, this is a DAE.

3.4. Solving the differential algebraic equations

The DAE for timestep m can be written as

$$K\mathbf{x}_m + M\ddot{\mathbf{x}}_m = F(\mathbf{x}_{m-1}). \tag{3.41}$$

DAEs generally have a singular mass matrix, which is also true for this system. Normally, a second order ODE can be solved by rewriting it into an first order ODE in the explicit form of $y' = f(y,t)$:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix}_m = \begin{bmatrix} 0 & I \\ -M^{-1}K & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}_m + \begin{bmatrix} 0 \\ M^{-1}F(\mathbf{x}_{m-1}) \end{bmatrix}. \tag{3.42}$$

But in DAEs with a singular mass matrix, this is not possible since the inverse of a singular matrix does not exist. There are different ways to tackle this issue in a numerical simulation of a DAE.

3.4.1. Naive approach

In the existing simulation software, the strategy to solve this system is the following. A system is created that incorporates this equation, but also the relationship between the displacement and velocity and the velocity and the acceleration using the Backward Euler scheme. By combining these relations, the system of equations that is created is non-singular:

$$\begin{bmatrix} K & 0 & M \\ I & -dt \cdot I & 0 \\ 0 & I & -dt \cdot I \end{bmatrix} \begin{bmatrix} \mathbf{x}_m \\ \dot{\mathbf{x}}_m \\ \ddot{\mathbf{x}}_m \end{bmatrix} = \begin{bmatrix} F(\mathbf{x}_{m-1}) \\ \mathbf{x}_{m-1} \\ \dot{\mathbf{x}}_{m-1} \end{bmatrix}. \tag{3.43}$$

This is referred to as the 'naive approach'. Note that the right-hand side vector has to be generated again for each timestep. Then in each timestep this system is solved to get $\mathbf{x}_m, \dot{\mathbf{x}}_m$ and $\ddot{\mathbf{x}}_m$.

3.4.2. Index reduction

Yet, the method for solving a DAE that is most found in literature is index reduction. The index of a DAE is the number of analytical differentiations needed for a system to transform it into an explicit ODE system for all of the unknowns [22]. So by analytical differentiations, the index of one DAE can be reduced. By differentiating the algebraic constraints twice with respect to t, we have

$$\ddot{u}_r = r\ddot{\alpha}_r, \tag{3.44}$$

$$\ddot{\alpha}_{r_1} = 0, \tag{3.45}$$

and the ODE

$$\begin{bmatrix} & \bar{K}_{f_1} & & & & \\ & & & \bar{K}_{f_2} & & \\ & & K_{T_{r_2}} & & & \end{bmatrix} \begin{bmatrix} \mathbf{u}_{r_1} \\ \mathbf{u}_{f_1} \\ \mathbf{u}_{r_2} \\ \mathbf{u}_{f_2} \\ \alpha_{r_1} \\ \alpha_{r_2} \end{bmatrix} + \begin{bmatrix} I & & & & -r\mathbf{1} & \\ & M_{f_1} & & & & \\ & & I & & & -r\mathbf{1} \\ & & & M_{f_2} & & \\ & & & & 1 & \\ & & M_{T_{r_2}} & & & -\frac{1}{2}m_r r^2 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_{r_1} \\ \ddot{\mathbf{u}}_{f_1} \\ \ddot{\mathbf{u}}_{r_2} \\ \ddot{\mathbf{u}}_{f_2} \\ \ddot{\alpha}_{r_1} \\ \ddot{\alpha}_{r_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.46}$$

is obtained. Therefore this is a DAE of index 2. The solution to this ODE can be approximated by using a suitable time integration scheme.

One advantage of the index reduction approach is that the right hand side of the system of equations is not dependent on the unknowns in the previous timestep. However, a significant disadvantage is that some information is lost when using Equation 3.44 instead of Equation 3.35. This will cause problems in some MOR techniques.

In addition, algebraic equations should hold as well in the reduced model [23] and reducing the algebraic equations would give unrealistic results. In the following approach the algebraic equations are eliminated before reducing, which would be more suitable when considering MOR.

3.4.3. Elimination of variables

Another way of solving a DAE is by elimination of variables. With Equation 3.35, we can describe u and $\ddot{u}$ on this roller and substitute this in Equation 3.36 and Equation 3.38. $\mathbf{u}_{r_1}$ is known explicitly and $\mathbf{u}_{r_2}$ is known implicitly as a function of α_{r_2}. With this approach, we have a system of $n_{f_1} + n_{f_2} - 1$ equations for $n_{f_1} + n_{f_2} - 1$ unknowns ($\mathbf{u}_{f_1} \in \mathbb{R}^{n_{f_1}-1}$, $\mathbf{u}_{f_2} \in \mathbb{R}^{n_{f_2}-1}$ and $\alpha_{roller_2} \in \mathbb{R}$).

The advantage of the elimination of variables is that a ODE is obtained with a non-singular mass and stiffness matrix. In this approach, however, the extra step of eliminating the algebraic variables might be time-consuming.

A general way of elimination of variables is comparable to static condensation [24]. Assume that the algebraic equations do not include first and second derivatives. Let $\mathbf{x}_f$ be the variables in the

system that will not be eliminated and $\mathbf{x}_e$ be the variables for which we have algebraic constraints and therefore will be eliminated. By reordering the equations and the unknowns, the system of equations can be written as

$$\begin{bmatrix} M_{ff} & M_{fe} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}}_f \\ \ddot{\mathbf{x}}_e \end{bmatrix} + \begin{bmatrix} K_{ff} & K_{fe} \\ K_{ef} & K_{ee} \end{bmatrix} \begin{bmatrix} \mathbf{x}_f \\ \mathbf{x}_e \end{bmatrix} = \begin{bmatrix} F_f \\ F_e \end{bmatrix}. \tag{3.47}$$

Then we can eliminate the variables $\mathbf{x}_e$ by expressing it as a function of $\mathbf{x}_f$ with

$$\mathbf{x}_e = K_{ee}^{-1}(F_e - K_{ef}\mathbf{x}_f), \tag{3.48}$$
$$\ddot{\mathbf{x}}_e = K_{ee}^{-1}(\ddot{F}_e - K_{ef}\ddot{\mathbf{x}}_f). \tag{3.49}$$

By substituting these into the dynamical system in Equation 3.47, it leads to the dynamical system for $\mathbf{x}_f$, given by

$$(M_{ff} - M_{fe}(K_{ee}^{-1}K_{ef}))\ddot{\mathbf{x}}_f + (K_{ff} - K_{fe}(K_{ee}^{-1}K_{ef}))\mathbf{x}_f = F_f - M_{fe}K_{ee}^{-1}\ddot{F}_e - K_{fe}K_{ee}^{-1}F_e. \tag{3.50}$$

After solving this system, an extra step of obtaining the eliminated unknowns has to be done using Equation 3.48. This eliminated system can be solved with a suitable time integration scheme, possibly even an explicit time-integration scheme. Note that, explicit time-integration should account for the stability conditions, which is hard for this simulation since the ODE does not represent the full system. Information on the interpolation step, that corrects for movement of the grid points after one timestep[2], should be included as well in the stability analysis for explicit time-integration.

As mentioned before, this elimination approach is preferred over the index reduction approach when using MOR, because the algebraic variables are treated separately. Consequently, these will not be reduced with MOR. However, while ideally the matrices in Equation 3.50 have to be inverted only once, small non-linearities may result in the matrices changing over time. Therefore, it can turn out that these matrices have to be computed every timestep again, which makes this approach computationally expensive.

3.4.4. Direct time integration

A widely used method for time integration in finite element analysis is the *Newmark-beta method* [25]:

$$\dot{\mathbf{x}}_{n+1} = \dot{\mathbf{x}}_n + \Delta t(1-\gamma)\ddot{\mathbf{x}}_n + \Delta t\gamma\ddot{\mathbf{x}}_{n+1}, \tag{3.51}$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t\dot{\mathbf{x}}_n + \Delta t^2\left(\frac{1}{2} - \beta\right)\ddot{\mathbf{x}}_n + \Delta t^2\beta\ddot{\mathbf{x}}_{n+1}, \tag{3.52}$$

where the subscript denotes it is the solution after timestep n or $n+1$. We should have $0 \leq \gamma \leq 1$ and $0 \leq 2\beta \leq 1$ and the choice $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$ is often taken because this leads to an unconditionally stable time integration. However, in the existing software for belt simulation, the *Backward Euler* time integration scheme is used [26] (see Equation 3.43). Therefore, we have decided to use this scheme as well in all approaches. The Backward Euler scheme is stable for stable systems.

For DAEs with a non-singular combination of M and $\Delta t^2 K$, the Backward Euler scheme can be applied directly, without any index reduction or elimination of variables. The Backward Euler scheme is obtained by substituting the following Backward Euler equations into the dynamical system,

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + \Delta t\dot{\mathbf{x}}_{n+1}, \\ \dot{\mathbf{x}}_{n+1} &= \dot{\mathbf{x}}_n + \Delta t\ddot{\mathbf{x}}_{n+1}. \end{aligned} \tag{3.53}$$

This leads to

$$(M + \Delta t^2 K)\ddot{\mathbf{x}}_{n+1} = F - K(\mathbf{x}_n + \Delta t\dot{\mathbf{x}}_n). \tag{3.54}$$

This is solved for $\ddot{\mathbf{x}}_{n+1}$ and with this, also the first derivative $\dot{\mathbf{x}}_{n+1}$ and the unknown $\mathbf{x}_{n+1}$ can be derived using Equation 3.53. For this system of equations, $M + \Delta t^2 K$ is non-singular, so Backward Euler could be applied directly. This is more efficient than the other approaches for handeling DAEs, therefore this will be used. However, the elimination approach will be considered as well when using MOR, because the algebraic equations should be treated separately from the differential equations in some MOR techniques.

[2] This interpolation step will be explained in section 3.5

Initial conditions

When a translation is set for one of the rollers, the belt has an initial displacement before starting the simulation. This initial displacement is obtained by solving $K\mathbf{x} = 0$ for $\mathbf{x}$. Furthermore, the initial velocity is zero except for the rotational speed of the drive roller and the velocity of the nodes on the drive roller according to Equation 3.34.

3.5. Interpolation step

Since the system of equations described in the preceding section is constructed based on equations that hold for displacement at grid points that are fixed in space and not fixed on the belt material, it becomes necessary to interpolate the displacement after each timestep onto these grid points. We call this process the interpolation step and it ensures that we can start each timestep with a comparable set of governing equations. The interpolation step is implemented as follows: the final displacement on

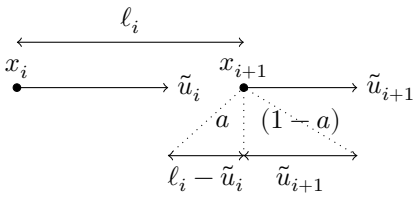


Figure 3.3: Visualization of the interpolation step for grid node $i+1$. $\tilde{u}$ denotes the temporary solution before interpolation.

a grid point $i+1$ after one timestep is a combination of the displacement of grid point i and grid point $i+1$, resulted from solving the system of equations described in the preceding section. It is assumed that the belt moves in the direction from grid point i to $i+1$. The resulting nodal displacement will be in reference to a moving grid that also is displaced by the mean displacement of the belt. Therefore, the mean displacement is subtracted from the resulting nodal displacement. This summarizes to

$$u_{i+1} = a_{i+1} \cdot \tilde{u}_i + (1 - a_{i+1}) \cdot \tilde{u}_{i+1} - \bar{\mathbf{u}}. \tag{3.55}$$

To ensure that grid points closer to the final displacement have a greater influence, we introduced a weight factor, denoted as a_{i+1}, with the value

$$a_{i+1} = \frac{\tilde{u}_{i+1}}{\ell_i - \tilde{u}_i + \tilde{u}_{i+1}}. \tag{3.56}$$

This weight factor contains values for intermediate displacements $\tilde{u}_{i+1}$ and $\tilde{u}_i$. Because the weight factor is multiplied by the intermediate displacement itself, a quadratic term appears. This makes the interpolation step non-linear.

By subtracting the mean in this interpolation step, the displacement in x will remain small. For every roller, the rotation in z direction (direction of the movement of the belt) will be 'interpolated' as well. Because it is only one value, this interpolation means that the mean, which is the value itself, is subtracted and the rotation is set to zero after each timestep.

It is important to note that the interpolation step relies on the assumption that a grid point at the current timestep does not move farther than the adjacent grid point from the previous timestep. In other words, the displaced grid point remains between its original position and the next grid point from the previous timestep. This ensures that the interpolation step only involves values between adjacent grid points, and no extrapolation is required. To ensure this behavior, we must select a timestep that satisfies the Courant-Friedrichs-Lewy (CFL) condition [27]. The CFL condition requires that the timestep is sufficiently small such that the displacement of any grid point within one timestep is less than the distance to the next grid point. If this condition is violated, the numerical domain of dependence would no longer align with the true physical domain of dependence, leading to instability or inaccurate results.

During the interpolation step, the mass is also transported. For one element i, a part of the mass of the previous element is added and also a part of the mass of the element is subtracted, which results in

$$m_i^p = m_i^{p-1} + a_{i-1} \cdot m_{i-1}^{p-1} - a_i \cdot m_i^{p-1}, \tag{3.57}$$

after timestep p. This becomes particularly significant when the displacement on the left grid point of an element is notably greater than the displacement on the right grid point. In such cases, the incoming mass to the element is higher the outgoing mass, resulting in an increase in the mass of that element. This increase then also has to be updated in the dynamical system, which introduces non-linearities in the mass matrix as well.

Due to this interpolation step, the non-linearities mentioned in this subsection appear. Additionally, we cannot directly relate the displacement of the nodes on the roller with the roller motion. Because of that, the governing algebraic and differential equations need to be recomputed after each timestep. This makes the modeling very inefficient. One solution would be to use floating frame of reference formulation. This is one of the most common approaches for modeling multibody systems. In [7], multibody dynamics is used to develop a finite element model of a conveyor belt, with the belt modeled as a one-dimensional string. Furthermore, model order reduction has been applied on simulations that use this approach by Nowakowski et al. [28]. Nonetheless, in this study we focus on this modeling choice as it is part of the existing software and remodeling is out of the scope of this research.

4

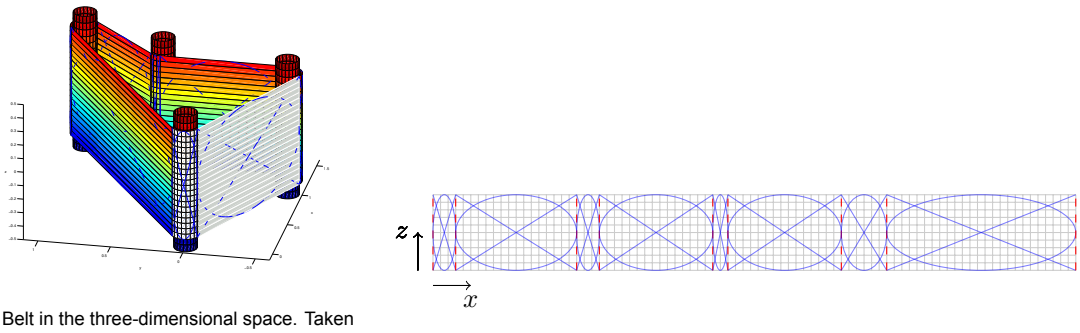
Extension to the two-dimensional model

In this section, we discuss the extension of the previous one-dimensional model to the two-dimensional model that is used in available simulation software [14]. The equations for the free-belt segments and roller segments are treated in section 4.1 and section 4.2, respectively. In section 4.3, the system of equations is assembled and in section 4.4, different methods for solving the resulting DAE are described. Finally, the differences regarding the interpolation step are explained in section 4.5.

This two-dimensional model can capture all translations and rotations in the three-dimensional space and provides a fuller description of the effect of misalignments of rollers. With this model, we can analyse both longitudinal and lateral displacement.

In the one-dimensional model, the belt is modeled as a string. In this two-dimensional model, the belt is modeled as a two-dimensional plane. This two-dimensional plane rotates around rollers that are defined in a three-dimensional space. Because of this, these rollers can translate and rotate around three dimensions. Therefore, first it is established how to project the belt plane in the three-dimensional space onto a two-dimensional plane. The result of the projection is shown in Figure 4.1. Note that the z direction is the same for the two-dimensional belt and three-dimensional space, but the x direction is not. Therefore, we will indicate x^{3D} as the x-coordinate in the three-dimensional space and x as the x-coordinate of the two-dimensional belt. For conveniency, we will do the same for y^{3D} and z^{3D}.

The belt is composed of free-belt segments and of roller segments. The first segment in the two-dimensional belt is the segment that is on roller 1 (the numbering of the rollers is defined as in chapter 2, and is in order of the belt path).



(a) Belt in the three-dimensional space. Taken from [14].

(b) Belt in a two-dimensional space.

Figure 4.1: Result from the projection of belt from a three-dimensional space to a two-dimensional space.

We have discretized the two-dimensional domain of the belt, which we will refer to as Ω. Belt segment i is indicated with Ω_i. The contactline left from Ω_i is $\partial\Omega_{i,l}$ and the contactline right from this segment is $\partial\Omega_{i,r}$. A free-belt segment is shown in Figure 4.2. One free-belt segment does not include the contactlines, marked in red. One roller segment Ω_i, shown in Figure 4.3, does include the contactlines. The x-coordinates of all nodes will not change over time and the interpolation step

will account for movement of grid points. The z-coordinate of all nodes can change over time and the z-coordinate will not be interpolated.

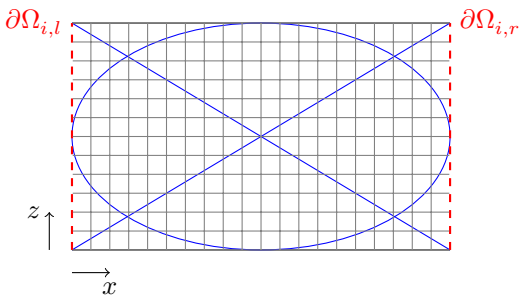


Figure 4.2: Free-belt segment Ω_i.

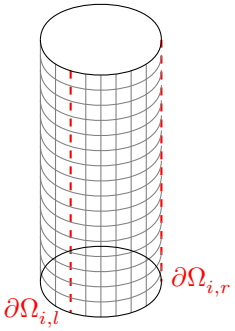


Figure 4.3: Roller segment Ω_i.

On one roller, both translational and rotational displacements can be defined. A translational displacement can also be combined with a rotational displacement to get a displaced roller. Both the types of displacements can be defined on all three directions that are depicted in Figure 2.3. The translational displacement will be defined by either setting a **translation** or a **force**. The rotational displacement can be defined by either setting a **rotation** or a **torque**. It is not possible to set a force/torque together with a translation/rotation in one direction. When there is neither a force/torque or a translation/rotation set, the roller is assumed to be fixed in that direction, so with a translation/rotation of zero. In this case, the values for force/torque can be nonzero and will be calculated during post-processing as an effect of the displacement resulting from the dynamical system.

4.1. Free-belt segments

On the free-belt parts, we will consider the belt to be under plane stress. It is assumed that the belt is not able to move in the transverse direction, perpendicular to the plane. Furthermore, there is no difference in displacement over the thickness of the belt. This is justified, since the thickness of the belt is very small related to the width and length of the belt. Therefore, we can assume that the normal stress and the shear stresses perpendicular to the plane are zero, which is crucial in applying the concept of plane stress [21].

The two-dimensional state of stress in one element is illustrated in Figure 4.4. The normal stresses σ_x and σ_z act in the x and z directions and the shear stresses τ_{xz} and τ_{zx} act on the edges of the element. Because the moments about the axis normal to the plane have to be summed to zero, we have $\tau_{xz} = \tau_{zx}$ (for a proof, see [21]). The three stresses are contained in the vector,

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_z \\ \tau_{xz} \end{bmatrix}. \tag{4.1}$$

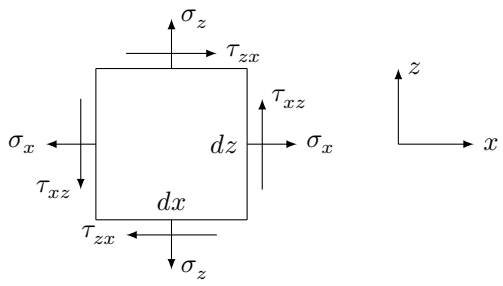


Figure 4.4: Two dimensional state of stress. Based on [21].

Hooke's law for two-dimensional plane stress is

$$\sigma = D\epsilon, \tag{4.2}$$

$$D = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}, \tag{4.3}$$

$$\epsilon = \begin{bmatrix} \epsilon_x \\ \epsilon_z \\ \gamma_{xz} \end{bmatrix}, \tag{4.4}$$

where ϵ is the strain on the element. The longitudinal strains are the changes in length per unit length of material fibers originally parallel to the x and z axes when the element undergoes deformation. The shear strain is the change in the original right angle made between dx and dz when the element undergoes deformation. The strain can be obtained from the derivatives of the displacement in the x and z direction with

$$\epsilon_x = \frac{\partial d_x}{\partial x}, \qquad \epsilon_z = \frac{\partial d_z}{\partial z}, \qquad \gamma_{xz} = \frac{\partial d_x}{\partial z} + \frac{\partial d_z}{\partial x}. \tag{4.5}$$

4.1.1. Meshing the domain

For the finite element analysis, a finite element discretization has to be defined on the two-dimensional belt. The belt is discretized into a rectangular grid of $n_\xi \times n_\eta$ grid points. The number of grid points in the z direction is equal over the whole length of the belt. The number of grid points in the x direction is defined per segment. There are grid points located exactly at the contactlines, represented in red in Figure 4.1b. Every grid point has an index (i, j) starting in the lower left corner. With lexicographical reordering of the indices we get an index $k = j + n_\eta(i-1)$ for each grid point. The lexicographical reordering is done such that the grid points over the width of the belt are consecutive. In this way, the segments can be separated clearly in the system of equations.

Using this discretization, rectangular grid elements are created, each with four connecting nodes, shown in Figure 4.5. The coordinates of the node k are represented by (x_k, z_k) or $(x_{i,j}, z_{i,j})$. Each node has two degrees of freedom, the displacement in the in x and z directions, notated as u_k and v_k, respectively ($u_{i,j}$ and $v_{i,j}$ are used as well for clarity in this chapter).

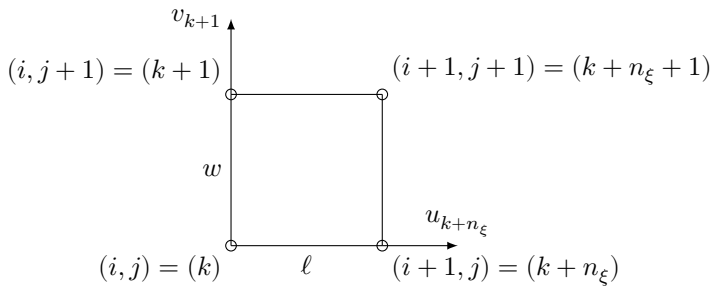


Figure 4.5: One grid element.

4.1.2. Construction of finite dimensional function spaces

For setting up the finite element problem, the basis functions for the finite element space have to be chosen. For every grid element there are four bilinear basis functions defined. For the element in Figure 4.5, the four basis functions are

$$N_{i,j}(x,z) = \frac{1}{lw}(x - x_{i+1,j})(z - z_{i,j+1}), \tag{4.6}$$

$$N_{i,j+1}(x,z) = -\frac{1}{lw}(x - x_{i+1,j+1})(z - z_{i,j}), \tag{4.7}$$

$$N_{i+1,j}(x,z) = -\frac{1}{lw}(x - x_{i,j})(z - z_{i+1,j+1}), \tag{4.8}$$

$$N_{i+1,j+1}(x,z) = \frac{1}{lw}(x - x_{i,j+1})(z - z_{i+1,j}). \tag{4.9}$$

With these four basis functions, we can describe a displacement at any interior point (x, z) in this element from the discrete displacement on the four nodes connected to one grid element with

$$\mathbf{u}(x,z) = \sum_{i,j} N_{i,j}(x,z) \cdot \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}, \tag{4.10}$$

or equivalently,

$$\mathbf{u}(x,z) = \begin{bmatrix} N_{i,j} & 0 & N_{i,j+1} & 0 & N_{i+1,j} & 0 & N_{i+1,j+1} & 0 \\ 0 & N_{i,j} & 0 & N_{i,j+1} & 0 & N_{i+1,j} & 0 & N_{i+1,j+1} \end{bmatrix} \begin{bmatrix} u_{i,j} \\ v_{i,j} \\ u_{i,j+1} \\ v_{i,j+1} \\ u_{i+1,j} \\ v_{i+1,j} \\ u_{i+1,j+1} \\ v_{i+1,j+1} \end{bmatrix}. \tag{4.11}$$

In Figure 4.6, the four basis functions are depicted. Note that the finite element basis functions satisfy

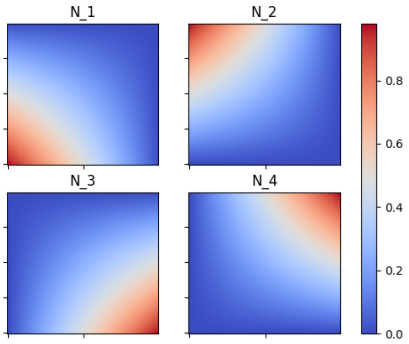


Figure 4.6: Basis functions per grid element.

the boundary Kronecker-Delta property. That is, every basis functions is 1 on the associated grid point and 0 on the other grid points. This Kronecker-Delta property assures that on the grid nodes, the displacement $\mathbf{u}(x_k, z_k)$ is the same as the discrete displacement on that grid point (u_k, v_k), obtained by the finite element solver. Finally, we have that $N_{i,j} + N_{i,j+1} + N_{i+1,j} + N_{i+1,j+1} = 1$ for every point in the grid element.

4.1.3. Assembly of equations matrices

Derive the stiffness matrix

We can describe the strain with the discrete displacement as

$$\epsilon_x = \frac{\partial d_x}{\partial x} = \sum_{i,j} \frac{\partial N_{i,j}}{\partial x}(x,z) \cdot u_{i,j}, \tag{4.12}$$

$$\epsilon_z = \frac{\partial d_z}{\partial z} = \sum_{i,j} \frac{\partial N_{i,j}}{\partial z}(x,z) \cdot v_{i,j}, \tag{4.13}$$

$$\gamma_{xz} = \frac{\partial d_x}{\partial z} + \frac{\partial d_z}{\partial x} = \sum_{i,j} \frac{\partial N_{i,j}}{\partial z}(x,z) \cdot u_{i,j} + \frac{\partial N_{i,j}}{\partial x}(x,z) \cdot v_{i,j}. \tag{4.14}$$

In matrix/vector expressions, this is equivalent with

$$\begin{bmatrix} \epsilon_x \\ \epsilon_z \\ \gamma_{xz} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial N_{i,j}}{\partial x} & 0 & \frac{\partial N_{i,j+1}}{\partial x} & 0 & \frac{\partial N_{i+1,j}}{\partial x} & 0 & \frac{\partial N_{i+1,j+1}}{\partial x} & 0 \\ 0 & \frac{\partial N_{i,j}}{\partial z} & 0 & \frac{\partial N_{i,j+1}}{\partial z} & 0 & \frac{\partial N_{i+1,j}}{\partial z} & 0 & \frac{\partial N_{i+1,j+1}}{\partial z} \\ \frac{\partial N_{i,j}}{\partial z} & \frac{\partial N_{i,j}}{\partial x} & \frac{\partial N_{i,j+1}}{\partial z} & \frac{\partial N_{i,j+1}}{\partial x} & \frac{\partial N_{i+1,j}}{\partial z} & \frac{\partial N_{i+1,j}}{\partial x} & \frac{\partial N_{i+1,j+1}}{\partial z} & \frac{\partial N_{i+1,j+1}}{\partial x} \end{bmatrix}}_{=:B} \begin{bmatrix} u_{i,j} \\ v_{i,j} \\ u_{i,j+1} \\ v_{i,j+1} \\ u_{i+1,j} \\ v_{i+1,j} \\ u_{i+1,j+1} \\ v_{i+1,j+1} \end{bmatrix}. \tag{4.15}$$

For all grid points, the principle of minimum potential energy is used to create an equation for the discrete displacements. This principle states that a system will be displaced in a way that minimized the potential energy. The potential energy is given by

$$P = U - F_{element}\mathbf{u}_{element}, \tag{4.16}$$

where U is the strain energy, $F_{element}$ the total force on an element and $\mathbf{u}_{element}$ the displacements on the nodes. The strain energy is given by

$$U = \frac{1}{2}\int_V \epsilon^\top \sigma dV, \tag{4.17}$$

with V the volume of the element. Because we have a constant thickness t_b of the element, we can get this out of the integral. Using this and Hooke's law as in Equation 4.2, we get

$$U = \frac{1}{2} t_b \int_{z_i}^{z_{i+1}} \int_{x_i}^{x_{i+1}} \epsilon^\top D\epsilon \; dx \; dz. \tag{4.18}$$

By the relation in Equation 4.15, the strain energy is expressed in terms of displacement like

$$U = \frac{1}{2} t_b \int_{z_i}^{z_{i+1}} \int_{x_i}^{x_{i+1}} \mathbf{u}_{element}^\top B^\top DB\mathbf{u}_{element} \; dx \; dz. \tag{4.19}$$

Because the terms in $\mathbf{u}_{element}$ are displacements on the nodes, and are therefore not dependent on x or z, they can be taken out of the integral, which results in

$$U = \frac{1}{2} t_b \mathbf{u}_{element}^\top \int_{z_i}^{z_{i+1}} \int_{x_i}^{x_{i+1}} B^\top DB \; dx \; dz \; \mathbf{u}_{element}. \tag{4.20}$$

As we are looking for the minimum of the potential energy, we want the derivative of P with respect to $\mathbf{u}_{element}$ to be zero. This results in

$$\frac{\partial P}{\partial \mathbf{u}} = \underbrace{\left[t_b \int_{z_i}^{z_{i+1}} \int_{x_i}^{x_{i+1}} B^\top D B \, dx \, dz \right]}_{=:K_{element}} \mathbf{u}_{element} - F_{element} = 0 \tag{4.21}$$

$$\Rightarrow \tag{4.22}$$

$$F_{element} = K_{element} \mathbf{u}_{element}. \tag{4.23}$$

Derive the mass matrix

Similar to the one-dimensional case, we have that Newton's second law of motion states that the total forces have to be equal to the mass times the acceleration. We likewise take the lumped mass for generating the mass matrix M_i, which is

$$m_i = \frac{\rho t_b w_i \ell_i}{4}, \tag{4.24}$$

$$F_i = K_i \mathbf{u}_i + M_i \ddot{\mathbf{u}}_i. \tag{4.25}$$

F_i is the eight-dimensional force vector of the forces in x and z direction on all the four connecting nodes,

$$F_i = \begin{bmatrix} F_{x,(i,j)} \\ F_{z,(i,j)} \\ F_{x,(i,j+1)} \\ F_{z,(i,j+1)} \\ F_{x,(i+1,j)} \\ F_{z,(i+1,j)} \\ F_{x,(i+1,j+1)} \\ F_{z,(i+1,j+1)} \end{bmatrix}. \tag{4.26}$$

4.2. Roller segments

The boundary condition is analogous to the one-dimensional case. The motion of the grid points on the rollers is described algebraically. The assumption is made that the belt sticks to the rollers, ensuring the displacement of the belt on the roller is a result of the displacement of the roller in the following way:

$$\dot{\mathbf{u}}_r = R_r \begin{bmatrix} \dot{\mathbf{d}}_r \\ \dot{\alpha}_r \end{bmatrix}. \tag{4.27}$$

When rotations are defined on rollers, the order of the rotations is of importance. First, the rotation in the x^{3D} direction is applied, then in the y^{3D} direction and finally in the z^{3D} direction. It can be verified that the resulting displacement is different when the order of the rotations is different.

The location of the grid points first have to be expressed in the three-dimensional space. After that, the rotations and displacements will be applied. Finally, the displacement on the two-dimensional belt will be calculated by subtracting the old location on the two-dimensional belt from the new location on the two-dimensional belt. All these operations are incorporated within the transformation R_r. Because we want a Dirichlet boundary condition as in the one-dimensional case, we discretize in time here first with Backward Euler to describe $\mathbf{u}_r(t)$ (based on Equation 3.35) with

$$\mathbf{u}_r(t) - R_r \begin{bmatrix} \mathbf{d}_r(t) \\ \alpha_r(t) \end{bmatrix} = \mathbf{u}_r(t - \Delta t) - R_r \begin{bmatrix} \mathbf{d}_r(t - \Delta t) \\ \alpha_r(t - \Delta t) \end{bmatrix}. \tag{4.28}$$

The matrix R_r also incorporates the length of the node to the center of the roller. This is for calculating the influence of the rotations in the three-dimensional space to a displacement on the two-dimensional belt. For a rotation in z^{3D} direction, this value is the radius of the node, so constant over the total

simulation (if the roller does not contain thickenings or other imperfections). But for a rotation in x^{3D} and y^{3D}, the value is dependent on the z coordinate of the node on the roller. This z coordinate will change if the belt moves in the z direction, which happens when misalignments are set or the belt is stretched (due to the effect of the Poisson's ratio). This causes the operation R_r to be non-linear. Note that, because the non-linearity is small, we could ignore this in the discretization step in time to obtain Equation 4.28.

4.2.1. Torque and force balance

Next to these algebraic equations, the equations for the torque and forces will be added to the system. When there is a force or torque set on the roller, we use the finite element approach to describe the equations that have to hold for this force or torque. Similar to the one-dimensional case, we only have a mass contribution to the forces at the elements on the rollers. This is the same lumped mass matrix as in Equation 4.24.

In the one-dimensional model, only a torque in one dimension can be set. In this case, however, we have a force in two dimensions on the belt, while seeking to satisfy a force in the three-dimensional space. So the two-dimensional force first has to be described in the three-dimensional space. After that, the force and torque in every dimension can be described with the sum of the nodal forces. For the force and torque in the x^{3D}-direction, this will be done in the following way:

$$F_{r,x} = \sum F_{x,k}, \tag{4.29}$$

$$T_{r,x} = \sum \mathbf{r}_{x,k} \times \begin{bmatrix} F_{y,k} \\ F_{z,k} \end{bmatrix}, \tag{4.30}$$

where $\mathbf{r}_{x,k}$ is the vector from grid point k to the rotational x axis of the roller, i.e. the absolute difference between the y^{3D}, z^{3D} coordinate of the node and the y^{3D}, z^{3D} coordinate of the rotational x axis. Note that this radius $\mathbf{r}_{x,k}$ can change over time when the y^{3D} or z^{3D} coordinate of the node changes. This can happen when the roller is misaligned in a different axis, but also when the belt is stretched, because of the Poisson effect. This is also true for $\mathbf{r}_{y,k}$ and causes small non-linearities.

For the nodal forces $F_{x,k}$ on the contactlines, the stiffness of the element on the adjacent free-belt segment has to be taken into account. As a result, the equations that describe the torque and force on one roller also involve displacements of the contactlines and adjacent free-belt nodes next to the accelerations of the nodes on the roller.

4.3. System of equations

Every displacement of the belt on the rollers is treated as an unknown in the system of equations. Next to these, the unknowns include all the displacements on the free-belt segments and the translations and rotations of the rollers. Each node in Ω has two degrees of freedom: u_k and v_k, in the x and z direction respectively. $\mathbf{u}$ is the displacement in x and z directions of all the grid nodes in Ω and is ordered alternately, where n_ξ is the number of grid points in x direction and n_η the number of grid points in the z direction. n is the total number of unknowns on the grid points and $\mathbf{u}$ is given by

$$\mathbf{u} = \begin{bmatrix} u_k \\ v_k \\ u_{k+1} \\ v_{k+1} \\ \vdots \\ u_{n_\eta} \\ v_{n_\eta} \\ \vdots \\ u_{n_\xi n_\eta} \\ v_{n_\xi n_\eta} \end{bmatrix} \in \mathbb{R}^n, \qquad n = 2n_\xi n_\eta. \tag{4.31}$$

Furthermore, we treat the rotations and translations (in three directions) on the rollers as unknowns. For every roller, that adds six unknowns to the system, indicated with

$$\chi_r = \begin{bmatrix} \mathbf{d}_r \\ \alpha_r \end{bmatrix} \in \mathbb{R}^6. \tag{4.32}$$

For p rollers, we have a total of $N = n + 6p$ unknowns:

$$\mathbf{x} = \begin{bmatrix} \mathbf{u} \\ \chi_{r_1} \\ \vdots \\ \chi_{r_p} \end{bmatrix}. \tag{4.33}$$

The system that includes the dynamics on the free-belt parts, the algebraic equations on the roller parts and the equations for rotation/translation of the rollers, for a system with three rollers is

$$\begin{bmatrix} I & \dots & & & & & R_{r_1} & & \\ & \bar{K}_{f_1} & & & & & & & \\ & \dots & I & \dots & & & & R_{r_2} & \\ & & & \bar{K}_{f_2} & & & & & \\ & & & \dots & I & \dots & & & R_{r_3} \\ & & & & & \bar{K}_{f_3} & & & \end{bmatrix} \begin{bmatrix} \mathbf{u}_{r_1} \\ \mathbf{u}_{f_1} \\ \mathbf{u}_{r_2} \\ \mathbf{u}_{f_2} \\ \mathbf{u}_{r_3} \\ \mathbf{u}_{f_3} \\ \chi_{r_1} \\ \chi_{r_2} \\ \chi_{r_3} \end{bmatrix}_m$$

$$+ \begin{bmatrix} 0 & \dots & & & & & & & \\ & M_{f_1} & & & & & & & \\ & \dots & 0 & \dots & & & & & \\ & & & M_{f_2} & & & & & \\ & & & \dots & 0 & \dots & & & \\ & & & & & M_{f_3} & & & \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_{r_1} \\ \ddot{\mathbf{u}}_{f_1} \\ \ddot{\mathbf{u}}_{r_2} \\ \ddot{\mathbf{u}}_{f_2} \\ \ddot{\mathbf{u}}_{r_3} \\ \ddot{\mathbf{u}}_{f_3} \\ \ddot{\chi}_{r_1} \\ \ddot{\chi}_{r_2} \\ \ddot{\chi}_{r_3} \end{bmatrix}_m = \begin{bmatrix} \mathbf{u}_{r_1}(t_{m-1}) - R_{r_1}\chi_{r_1}(t_{m-1}) \\ F_{f_1} \\ \mathbf{u}_{r_2}(t_{m-1}) - R_{r_2}\chi_{r_2}(t_{m-1}) \\ F_{f_2} \\ \mathbf{u}_{r_3}(t_{m-1}) - R_{r_3}\chi_{r_3}(t_{m-1}) \\ F_{f_3} \end{bmatrix}.$$

The equations for the translation and rotation of each roller will be added to this system. This is a dynamical equation if the force or torque is prescribed and an algebraic (trivial) equation if the translation or rotation is set to be fixed or static. Each roller is subject to six equations. When the equation for the translation or rotation is trivial, the given translation or rotation is put in F.

4.4. Solving the differential algebraic equations

In section 3.4, different methods for solving a DAE are described. Because the system matrices have the same structure, the different approaches can be applied to the DAE for the two-dimensional case as well. Hence, we use direct time integration with Backward Euler here as well. Elimination of variables will also be considered when applying MOR.

4.4.1. Initial conditions

In case the rollers are misaligned, the belt has an initial displacement. For this initial displacement, the following system of equations is solved:

$$K\mathbf{x} = F. \tag{4.34}$$

We assume $\mathbf{u}_{r_i}(t_{-1}) - R_{r_i}\chi_{r_i}(t_{-1}) = 0$. With that, we have in the initial condition that the displacement of the belt on the rollers has to be exactly equal to the displacement caused by the rotations and displacements on the rollers. Note that in F, the values for the static rotation and translation are given in the last $6p$ entries in F.

There is no belt velocity/acceleration involved yet in the displacements of the free-belt, so it is assumed that a dynamical simulation starts with a free-belt without a velocity. The belt on the idle rollers and the idle rollers itself have no initial velocity as well. The belt on the drive roller and the drive roller itself have an initial velocity. These are taken to be as in Equation 4.27, where $\dot{\alpha}_r$ is given.

For DAEs, it is necessary to have *consistent* initial conditions [29]. This means that the initial conditions should satisfy the algebraic constraints that are present in the DAE. In this case, the DAE is satisfied because we use Equation 4.34 to calculate the initial conditions.

4.5. Interpolation step

The interpolation step is similar to the interpolation step in the one-dimensional case. For this two-dimensional system, the displacement in the x direction is interpolated. The z displacement will not be interpolated. Next to that, the displacements of the nodes on rollers is interpolated differently as the displacements of the nodes on free-belt segments.

The interpolated displacement is a result of the displacement in the horizontally adjacent grid point and the grid point itself. Whether the left or the right node is taken as adjacent grid point depends on the direction of displacement. The interpolation step is visualized in Figure 4.7. In this interpolation step, the mass is updated as well and the roller rotation in z^{3D}-direction (not in the x^{3D} and y^{3D} directions) is interpolated in a similar way as it is done in the one-dimensional model. As explained in section 3.5, extrapolation is not possible in the interpolation step. Therefore, the timestep is chosen to ensure that the CFL condition is satisfied.

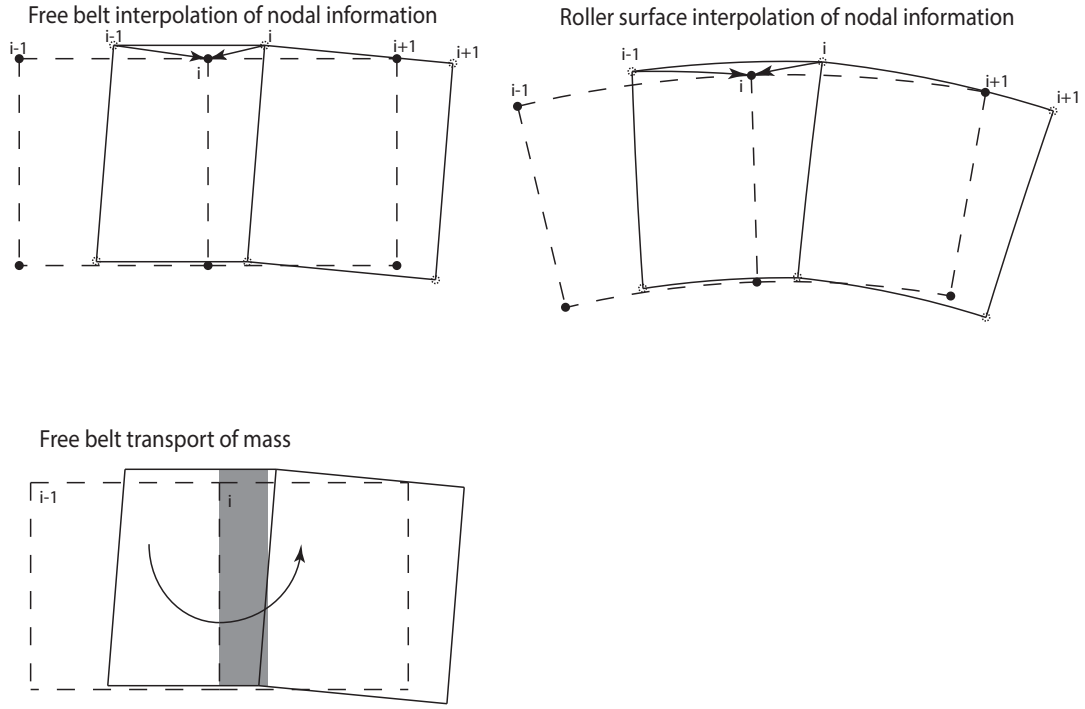


Figure 4.7: Interpolation for a free-belt (upper-left) and on the radius-surface (upper-right) and transfer of mass (lower) for the interpolation phase after one time-step. Taken from [14].

In section 4.2, we explained that the operation R_r in the stiffness matrix contains small non-linearities. In addition, the interpolation step brings non-linearities into the system. First of all, the interpolation of nodal information is based on a comparable weight factor as in section 3.5. This introduces a quadratic term in the update operation. Furthermore, the element mass is transported and therefore not constant for the dynamic simulation. Therefore, the mass matrix also contains non-linearities.

In this chapter, we have shown how the two-dimensional model is set up. The structure of the equations and the modeling steps in the two-dimensional case are analogous to those for the simple one-dimensional case derived in the previous chapter. Therefore, we will apply MOR on the two-dimensional case in the same way as we would in the one-dimensional case. We expect comparable characteristics of the different MOR methods when applying them on the one- and two-dimensional models.

5

Model order reduction

This chapter introduces methods for reducing the dynamical system that describes the movement of a one- and two-dimensional belt. In section 5.1, general projection based model order reduction is explained of which two projection based methods are outlined, modal decomposition and POD. In section 5.2, DMD is explained. The three MOR methods mentioned in this chapter will be applied on the system of equations. The methods are proposed for linear systems of equations. However, as seen in chapter 3 and chapter 4, the system is nonlinear because of the weakly nonlinear governing equations and the nonlinear interpolation step. Except, since we expect that our conveyor belt system becomes weakly nonlinear, we will focus on linear MOR.

As discussed in the introduction, non-intrusive MOR methods operate by constructing a ROM while not requiring explicit knowledge of the original model equations. Therefore, non-intrusive MOR methods use data obtained from high-order simulations or experiments (data-driven). These methods are particularly useful when the system's dynamics are highly complex or when the governing equations are difficult to access or modify. Some example methods of non-intrusive MOR are Dynamic Mode Decomposition [30], Deep Convolutional Autoencoders [31] and methods using neural networks [32]. Dynamic Mode Decomposition is intended for linear systems. An autoencoder can also be used to describe the states of non-linear systems in a reduced setting. With neural networks, both non-linear and linear dynamical systems can be learned. It is also possible to include known information of the system when using neural networks, with physics-informed neural networks [33].
On the other hand, intrusive MOR methods involve directly modifying the governing equations of the original model to derive a ROM. For this reduction, we can use properties of the underlying equations of the system (model-driven) or use simulation data (data-driven). While intrusive methods may require additional effort to implement, they can offer significant advantages in terms of accuracy because detailed knowledge of the system dynamics is used, which is not the case for non-intrusive MOR methods. Some examples are Proper Orthogonal Decomposition [34] (data-driven), modal decomposition [35] (model-driven) and balanced truncation [36] (model-driven).

We want to reduce the model for different misalignments of rollers. The misalignments can be seen as parameters that determine the dynamical system. For parametric systems, intrusive parametric model order reduction techniques have been developed [37]. These techniques provide a ROM for the full set of parameters. Once the model is reduced, the result for all the different parameters can be computed efficiently with this reduced model. The reduction of this system can be extended to parametric model order reduction when model order reduction appears to be not efficient enough, but this will be outside of the scope of this research.

For selecting either a non-intrusive or intrusive method, the computational effort of simulations and prior knowledge are considered. In this case, where simulations are computationally expensive and there exists prior knowledge of the dynamical system, we will start by employing intrusive MOR techniques. By directly modifying the governing equations, intrusive methods can preserve important system properties and ensure the accuracy and robustness of the reduced order model. Next to that, there is no need for results from computationally expensive simulations. However, when we want to apply MOR on the full simulation, not only on the dynamical system step, we still have a part of the simula-

tion where the underlying equations are considered unknown, namely the interpolation step. For this reason, also non-intrusive MOR will be investigated.

5.1. Projection-based model order reduction

Projection-based methods in MOR aim to construct a reduced-order model by projecting the high-dimensional system onto a lower-dimensional subspace [38]. This subspace captures the dominant behavior of the system while discarding less significant dynamics. The reduced-order model is typically expressed in terms of a low-dimensional state vector $\mathbf{z}(t) \in \mathbb{R}^r$, where r is the desired reduced order. First, we will introduce a general projection-based MOR method. Then, two methods to find the projection are described.

Let us consider a high-dimensional system described by the equations

$$K\mathbf{x} + M\ddot{\mathbf{x}} = F, \tag{5.1}$$

where $K \in \mathbb{R}^{N\times N}$ and $M \in \mathbb{R}^{N\times N}$ are the stiffness and mass matrices, respectively, $\mathbf{x}(t) \in \mathbb{R}^N$ represents the state vector and $F \in \mathbb{R}^N$ denotes the time-dependent external forcing. Projection-based MOR involves finding a transformation matrix $V \in \mathbb{R}^{N\times r}$ that matches the high-dimensional state $\mathbf{x}(t)$ to a low-dimensional space via a linear transformation V. This projection is represented as

$$\mathbf{x}(t) \approx V\mathbf{z}(t), \tag{5.2}$$

where $\mathbf{z}(t) \in \mathbb{R}^r$ is the reduced state vector. Substituting $\mathbf{x}$ from Equation 5.2 into Equation 5.1, we obtain the system for the reduced state:

$$KV\mathbf{z} + MV\ddot{\mathbf{z}} = F. \tag{5.3}$$

However, the system in Equation 5.3 is typically overdetermined for $\mathbf{z}$, as it has N equations and only r unknowns. To address this, we enforce orthogonality to some space spanned by the columns of a matrix $W \in \mathbb{R}^{N\times r}$. This leads to

$$W^\top KV\mathbf{z} + W^\top MV\ddot{\mathbf{z}} = W^\top F. \tag{5.4}$$

The reduced-order model is obtained by defining

$$\begin{aligned}\hat{K} &:= W^\top KV,\\ \hat{M} &:= W^\top MV,\\ \hat{F} &:= W^\top F.\end{aligned}$$

When $W = V$ and V is orthonormal, this is called the Galerkin projection, otherwise it is the more general Petrov-Galerkin projection [38].

5.1.1. Modal decomposition

In the context of structural dynamics and finite element analysis, modal decomposition is often employed to identify the characteristic modes of vibration of a mechanical structure [39]. These modes represent the natural oscillations within the structure and can be used to understand its dynamic response to external forces.

Modal decomposition identifies the characteristic modes of a system by solving the generalized eigenvalue problem of the free vibration problem of Equation 5.1 given by

$$(K - \omega_i^2 M)\phi_i = 0, \tag{5.5}$$

where the eigenvalue $\lambda_i = \omega_i^2$ is one natural frequency of the system and ϕ is the eigenvector (or eigenmode). When the slowly varying dynamics of the system are considered, the response can be accurately approximated by r low frequency modes [39].

$$\mathbf{x}(t) \approx \sum_{i=1}^{r} \phi_i z_i(t) = \Phi\mathbf{z}(t). \tag{5.6}$$

Given that $r << N$, we achieve a reduced state $\mathbf{z}(t)$ through a Petrov-Galerkin projection where $V = W = \Phi$. Note that Φ is not orthonormal, but generally is constructed to be M-orthogonal, that is

$$\phi_j M \phi_i = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases} \tag{5.7}$$

5.1.2. Proper Orthogonal Decomposition

POD is a widely used technique for MOR that aims to capture the dominant modes of a dynamical system [34]. Given a set of snapshots $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$, where each $\mathbf{x}_i \in \mathbb{R}^N$ represents the state at a particular time. These snapshots are arranged in a snapshot matrix

$$X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m] \in \mathbb{R}^{N \times m}. \tag{5.8}$$

The key step in POD is to compute the largest r eigenvalues with their corresponding eigenvectors of the correlation matrix $X^\top X$. Or, equivalently, we can compute the singular value decomposition (SVD), where Σ contains the singular values in decreasing order. The SVD of X is

$$X = U \Sigma V^\top, \tag{5.9}$$

$$U \in \mathbb{R}^{N \times N}, \quad \Sigma \in \mathbb{R}^{N \times m}, \quad V \in \mathbb{R}^{m \times m}. \tag{5.10}$$

The first r columns of U capture the most dominant variations in the data and will therefore be used as POD basis. Usually, the dimension r of the lower-dimensional subspace will be chosen after the computation of the SVD. This is because the singular values give an impression of how much of the total energy is contained in the basis spanned by the corresponding singular vectors. The goal is to choose r such that the energy contained in the basis, defined by the following statement, is close to one:

$$E(r) = \frac{\sum_{j=1}^{r} \sigma_j^2}{\sum_{j=1}^{m} \sigma_j^2}. \tag{5.11}$$

After truncating the SVD with r we get

$$X \approx \tilde{U} \tilde{\Sigma} \tilde{V}^\top, \tag{5.12}$$

$$\tilde{U} \in \mathbb{R}^{N \times r}, \tilde{\Sigma} \in \mathbb{R}^{r \times r}, \tilde{V} \in \mathbb{R}^{m \times r}. \tag{5.13}$$

Now, the POD basis is defined as $\tilde{U}$ and the reduced system is obtained by taking $V = W = \tilde{U}$ in Equation 5.4.

The process of selecting snapshots is fundamental to the accuracy and efficiency of POD. It involves determining the subset of system states or configurations that adequately represent the system's behavior over time and space.

5.2. Dynamic Mode Decomposition

Contrary to the previous methods, this method is equation free. There is no need to project a known equation onto a reduced space, it is fully based on the gathered input data [30]. This method uses the same snapshot matrix as the one used in POD. For this snapshot matrix, however, it is important that this matrix is ordered. Each pair of consecutive snapshots should be separated by the same timestep Δt and assembled like

$$X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m] \in \mathbb{R}^{N \times m}. \tag{5.14}$$

To apply DMD, the assumption is made that the state after one timestep of Δt is obtained by a linear map A on the previous state. We assume here that this linear map approximates the operation in the full simulation with

$$\mathbf{x}_{i+1} \approx A \mathbf{x}_i. \tag{5.15}$$

By introducing the succeeding sequence of snapshots in the snapshot matrix

$$X' = [\mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_{m+1}] \in \mathbb{R}^{N \times m}, \tag{5.16}$$

it follows

$$X' \approx AX. \tag{5.17}$$

Dynamic Mode Decomposition identifies the best approximation of the linear map A. This is done with the use of the pseudo-inverse of X, where the compact singular value decomposition is used, given by

$$X = U\Sigma V^\top, \tag{5.18}$$

$$U \in \mathbb{R}^{N \times m}, \Sigma \in \mathbb{R}^{m \times m}, V \in \mathbb{R}^{m \times m}. \tag{5.19}$$

Because normally, the dimension of X is large and to be reduced, we use the truncated singular value decomposition, given by

$$X \approx \tilde{U}\tilde{\Sigma}\tilde{V}^\top, \tag{5.20}$$

$$\tilde{U} \in \mathbb{R}^{N \times r}, \tilde{\Sigma} \in \mathbb{R}^{r \times r}, \tilde{V} \in \mathbb{R}^{m \times r}. \tag{5.21}$$

The operator A in DMD is approximated by

$$A \approx X'X^+ = X'\tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^\top. \tag{5.22}$$

We reduce this operator by projecting it onto the left singular values of X with

$$\tilde{A} = \tilde{U}^\top A\tilde{U} = \tilde{U}^\top X'\tilde{V}\tilde{\Sigma}^{-1}. \tag{5.23}$$

Once $\tilde{A}$ is determined, the DMD modes and associated dynamics can be extracted. By Equation 5.15, we calculate $\mathbf{x}_k$ with the use of A^k. This can be efficiently calculated by determining the eigenvalues and eigenvectors of A: Φ and Λ. Then, we can say

$$\mathbf{x}_k \approx \Phi\Lambda^k\Phi^{-1}\mathbf{x}_0. \tag{5.24}$$

The eigenvalues are obtained by computing the eigenvectors and eigenvalues of $\tilde{A}$, that is

$$\tilde{A}W = W\Lambda. \tag{5.25}$$

When W, Λ are the eigenvectors and values of $\tilde{A}$, we know that Φ, Λ are eigenvectors and values of A. Φ is calculated as done in the exact DMD approach demonstrated by Tu [40]. In this approach,

$$\Phi = X'\tilde{V}\tilde{\Sigma}^{-1}W. \tag{5.26}$$

To see that Φ are eigenvectors of A, we write out $A\Phi$ as

$$A\Phi = AX'\tilde{V}\tilde{\Sigma}^{-1}W = X'\tilde{V}\tilde{\Sigma}^{-1}\underbrace{\tilde{U}^\top X'\tilde{V}\tilde{\Sigma}^{-1}}_{\tilde{A}}W \tag{5.27}$$

$$= X'\tilde{V}\tilde{\Sigma}^{-1}\tilde{A}W = X'\tilde{V}\tilde{\Sigma}^{-1}W\Lambda = \Phi\Lambda. \tag{5.28}$$

Note that these are not all the eigenvalues and vectors of A, because $\tilde{A}$ is of a smaller size.

For calculating the trajectory with Equation 5.24, we cannot use Φ^{-1}, because it is not square. Next to that, it is not preferred to use the pseudo-inverse, because it can be of high dimension. Therefore we use the lower dimensional approximation $\mathbf{x}_0 = \tilde{U}\mathbf{v}_0$. And we use a vector $\mathbf{b}$ that can approximate $\Phi^{-1}\mathbf{x}_0$, that is

$$\begin{aligned}
\mathbf{x}_0 &= \Phi\mathbf{b} \\
\tilde{U}\mathbf{v}_0 &= X'\tilde{V}\tilde{\Sigma}^{-1}W\mathbf{b} \\
\tilde{U}^\top\tilde{U}\mathbf{v}_0 &= \underbrace{\tilde{U}^\top X'\tilde{V}\tilde{\Sigma}^{-1}}_{\tilde{A}}W\mathbf{b} \\
\mathbf{v}_0 &= \tilde{A}W\mathbf{b} \\
\mathbf{v}_0 &= W\Lambda\mathbf{b} \\
\mathbf{b} &= (W\Lambda)^+\mathbf{v}_0 = (W\Lambda)^+\left(\tilde{U}^\top\mathbf{x}_0\right).
\end{aligned} \tag{5.29}$$

To put all things together, we can describe the trajectory of $\mathbf{x}$ by

$$\mathbf{x}_k \approx \Phi \Lambda^k \mathbf{b} = \sum_{i=1}^{r} b_i \phi_i \lambda_i^k. \tag{5.30}$$

When we write $\lambda = e^{\omega \Delta t}$, we have

$$\mathbf{x}_k \approx \sum_{i=1}^{r} b_i \phi_i e^{\omega k \Delta t} = \sum_{i=1}^{r} \underbrace{b_i \phi_i}_{=:\theta_i} e^{\omega t}. \tag{5.31}$$

θ_i are the scaled modes and describe the behaviour of the state over time. The selection of modes is not unique [41]. We will examine the selection of modes for this specific application in the following chapters.

In some cases, it can happen that $\mathbf{x}_0$ is not a correct representation of the state over time. Hence, it can be inconvenient to use $\mathbf{b}$ in the scaled mode. Alternatively, we can describe the trajectory starting at time $t_0 = k_0 \Delta t$ with

$$\mathbf{x}_k \approx \Phi \Lambda^{k-k_0} \tilde{\mathbf{b}}, \tag{5.32}$$

where we compute $\tilde{b}$ to satisfy $\mathbf{x}_{k_0} \approx \Phi \tilde{\mathbf{b}}$ with the same method as in Equation 5.29. When we write $\lambda = e^{\omega \Delta t}$, we have

$$\mathbf{x}_k \approx \sum_{i=1}^{r} \tilde{b}_i \phi_i e^{\omega (k-k_0) \Delta t} = \sum_{i=1}^{r} \tilde{b}_i \phi_i e^{\omega (t-t_0)}. \tag{5.33}$$

DMD offers several advantages, including its data-driven nature, which allows it to be applied directly to observational data without the need for detailed knowledge of the underlying system dynamics. This makes DMD particularly useful when system models are unavailable or when dealing with highly complex and nonlinear systems. Additionally, DMD provides a clear and interpretable representation of the system's dominant coherent structures and their associated temporal behaviors, making it a valuable tool for understanding complex dynamical systems. However, DMD also has some limitations. For example, it assumes linear dynamics between consecutive snapshots, which may not always hold true for nonlinear systems. Moreover, DMD can be sensitive to noise present in the data, which may affect the accuracy of the extracted modes and temporal dynamics.

6

Numerical results for the one-dimensional model

We first apply model order reduction methods to the simplified one-dimensional model derived in chapter 3. In section 6.1, the performance measures are introduced and in section 6.2 to section 6.4, details on the implementations are explained and results are shown.

The one-dimensional model is implemented from scratch and all characteristics of the underlying equations are known. The results when applying model order reduction on this model are theoretically substantiated with the knowledge about the model. In the next chapter, we will show whether the expectation based on these results applies in the two-dimensional belt model as well.

We will apply the three model order reduction methods explained in chapter 5 on this model. The model is divided into two parts. Each timestep, first the dynamical system is solved and after that the interpolation step is executed. Because the interpolation step is not a conventional way of modeling and not fully known for the two-dimensional model, we will treat this interpolation step as a *black-box*. That means, we do not know anything about this step except that we can execute it.

Modal decomposition is fully based on the governing equations of the system, POD is based on the governing equations in combination with simulation data and DMD is fully based on simulation data. Because we have treated the interpolation step as a black-box, we cannot include that in modal decomposition and POD. Therefore modal decomposition and POD are only applied on the first part of each timestep, that is, solving the dynamical system. DMD can include both solving the dynamical system and the interpolation step in the reduction because it is not based on any of the equations, only the simulation data.

When reducing the dynamical system with intrusive MOR, we have to be aware that the dynamical system in this case consists of both algebraic equations and differential equations. Benner and Stykel [15] have discussed different MOR methods for differential algebraic equations. Most methods rely on separating the dynamic with the algebraic variables. The algebraic equations are the constraints in the system. These constraints should also be satisfied exactly in the reduced model [23]. Therefore we can only reduce the dynamic part of the system. We will first eliminate the algebraic equations with the procedure as explained in section 3.4. This results in a system of ordinary differential equations.

6.1. Performance measures

We use the Root Mean Squared Error (RMSE) and the Relative Root Mean Squared Error (RRMSE) to compare two trajectories. Let $\mathbf{x}$ and $\tilde{\mathbf{x}}$ be the vectors of the actual and approximated states, respectively. When we have N unknowns, the RMSE is defined as

$$RMSE(\mathbf{x}, \tilde{\mathbf{x}}) = \left(\frac{1}{N} \sum_{i=1}^{N} |x_i - \tilde{x}_i|^2 \right)^{1/2}. \tag{6.1}$$

Because this has a unit of millimeters, it is easy to see how significant the error is. Next to that, it has the advantage not to cancel errors out, as is the case, for example, with the bias.

The RRMSE, in literature sometimes referred to as the Normalized Root Mean Squared Error [42], is defined as

$$RRMSE(\mathbf{x}, \tilde{\mathbf{x}}) = \left(\sum_{i=1}^{N} |x_i - \tilde{x}_i|^2 / \sum_{i=1}^{N} |x_i|^2 \right)^{1/2}. \tag{6.2}$$

The RRMSE normalizes the error, which makes it easier to compare the accuracy of different models and to see how large the error relative to the solution is. However, the disadvantage is that when the actual state is small in norm, the RRMSE can blow up.

6.1.1. Computational speed

To evaluate the effectiveness of the MOR methods, it is also necessary to compare the computational speed. In addition to solving the system, two extra processes are executed that are dominant in terms of computational speed.

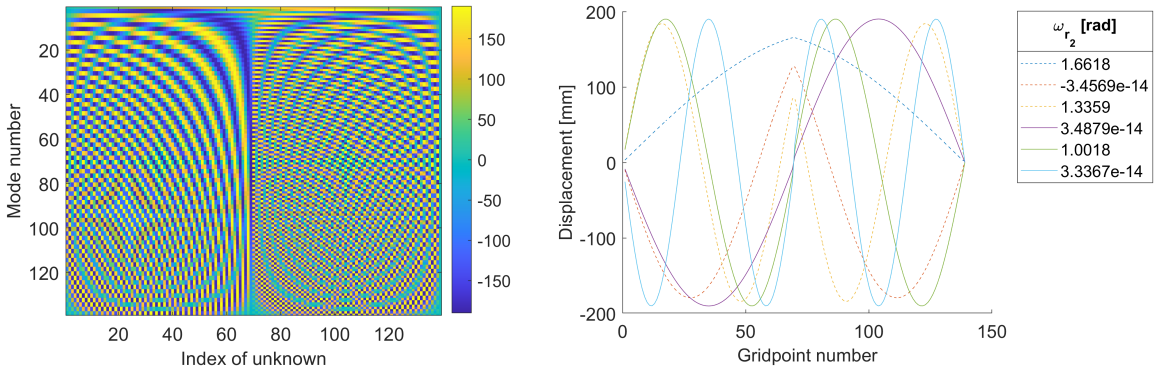
- **Update system equations**: firstly, we have the construction of the system, which must be performed at each timestep due to the changing governing equations.
- **Interpolation**: secondly, the interpolation step will be conducted at the end of each timestep. This is only reduced with non-intrusive MOR.
- **Solve system**: lastly, we have the step of solving the system, which is the step that will be reduced by all MOR techniques.

The first two steps are not expected to have a speedup when applying intrusive MOR, as intrusive MOR cannot be applied to processes where the computations are treated as a black-box. For the one-dimensional setup, a single simulation will be done, and the mean computational time across all time steps will be used. In the two-dimensional setup, the mean computational time will be derived from all time steps of five simulations.

6.2. Modal decomposition

We will start by reducing the dynamical system with modal decomposition. As explained in section 3.4, for reducing the system, the algebraic equations are eliminated. This resulting dynamical system in Equation 3.50 consists of a set of ordinary differential equations. Note as well, that this reduction is based on the generalized eigenvalues resulting from Equation 5.5. The eigenvalues are a indication of the frequency of the corresponding mode. When the reduction is applied on differential-algebraic equations, it can result in eigenvalues with infinite value, because we have a singular mass matrix. This indicates that some modes have infinite frequency, which is a non-physical result and therefore gives another reason to eliminate the algebraic variables from the governing equations before applying modal decomposition.

We will demonstrate modal decomposition by considering the example system configuration that is also evaluated in chapter 3 and Figure 3.1. We use 40 nodes on the roller segments and 70 nodes on the free-belt segments. By adding the translations and rotations of the four rollers, this results in a dynamical system with $N = 224$. After eliminating the algebraic variables (the displacements of the roller nodes and the given translations and rotations) the number of unknowns in the system is decreased to 139. The modes that are calculated with Equation 5.5 are depicted in Figure 6.1. These modes are representations of the solution in the eliminated system in Equation 3.50. The algebraic variables are eliminated here, so are not included in the plot for the modes. The modes are therefore only the values for displacement on the free-belt segment and the rotation of roller two. In Figure 6.1a, each row represents a mode. The modes are ordered with frequency: the mode with the lowest mode number has the lowest frequency. The modes appear to be composed of two parts by the clear seperation of the figure into two parts at unknown with index 69. The two parts are the displacements of the nodes on the two free-belt segments. The last column (column 139) represents the rotation of roller two. This is a different physical quantity as the displacement values (the rest of the non-eliminated unknowns). In Figure 6.1b, the rotation is therefore listed in the legend separately. This figure shows the displacement



(a) All full modes. The mode number indicates the frequency order (a low mode number corresponds to a low frequency).

(b) Displacement of six modes with the lowest frequency with corresponding roller rotation r_2. Even modes are shown with a solid line and odd modes with a dashed line.

Figure 6.1: Modes resulting from modal decomposition for a system with 40 elements on one roller segment and 70 elements on one free-belt segment.

of the six modes with the lowest frequency.

In Figure 6.1b, we have made a distinction between the odd and the even modes. The odd modes seem to have more influence in the reduced order model than the even modes. The reason for this is that the crucial information in the right-hand side is not visible in the even modes. When eliminating algebraic variables, we have in the first equation on the first free-belt segment a term with $u_{n_{r_1}+1}$. This term can be found in subsection 3.1.5 and is

$$K[n_{r_1}+2, n_{r_1}+1] = -\frac{EA}{\ell_{n_{r_1}+1}}. \tag{6.3}$$

For the last equations on the first free-belt segment, we have a term with $u_{n_{r_1}+n_{f_1}}$ which is

$$K[n_{r_1}+n_{f_1}, n_{r_1}+n_{f_1}+1] = -\frac{EA}{\ell_{n_{r_1}+n_{f_1}}}. \tag{6.4}$$

On the second free-belt segment, we have comparable values, given by

$$K[n_{r_1}+n_{f_1}+n_{r_2}+2, n_{r_1}+n_{f_1}+n_{r_2}+1] = -\frac{EA}{\ell_{n_{r_1}+n_{f_1}+n_{r_2}+1}}, \tag{6.5}$$

$$K[n,1] = -\frac{EA}{\ell_n}. \tag{6.6}$$

These displacements are for grid points on one roller segment and are therefore algebraic and determined by the solution in the previous equation and the roller rotation as

$$u_i = u_i(t-\Delta t) - r\alpha_{r_1}(t-\Delta t) + \alpha_{r_1}(t) \text{ for } i = 1, n_{r_1}+1, \tag{6.7}$$

$$u_i = u_i(t-\Delta t) - r\alpha_{r_2}(t-\Delta t) + \alpha_{r_2}(t) \text{ for } i = n_{r_1}+n_{f_1}+1, n_{r_1}+n_{f_1}+n_{r_2}+1. \tag{6.8}$$

When we substitute the algebraic variables into the dynamic equations, the terms appear in the right-hand side. The variable $\alpha_{r_2}(t)$ is not substituted, because this is not an algebraic variable since it is determined by the torque balance of roller 2. This results in a right-hand side

$$F = \begin{bmatrix} \frac{EA}{\ell_{f_2}} \cdot \left(u_{n_{r_1}+1}(t-\Delta t) - r\alpha_{r_1}(t-\Delta t) + \alpha_{r_1}(t)\right) \\ \mathbf{0} \\ \frac{EA}{\ell_{f_1}} \cdot \left(u_{n_{r_1}+n_{f_1}+1}(t-\Delta t) - r\alpha_{r_2}(t-\Delta t)\right) \\ \frac{EA}{\ell_{f_1}} \cdot \left(u_{n_{r_1}+n_{f_1}+n_{r_2}+1}(t-\Delta t) - r\alpha_{r_2}(t-\Delta t)\right) \\ \mathbf{0} \\ \frac{EA}{\ell_{f_2}} \cdot \left(u_1(t-\Delta t) - r\alpha_{r_1}(t-\Delta t) + \alpha_{r_1}(t)\right) \\ \dots \end{bmatrix}. \tag{6.9}$$

When projecting the right-hand side F onto the space spanned by the columns of Φ, only the vectors that have some resemblance with F, appear in the projection. In this example setting, the second roller eventually rotates with the same speed as the first (drive) roller. Hence, also, the displacements on these rollers will eventually have approximately the same value. In that case, when looking at Equation 6.9, the first and second to last value will be approximately the same and the two middle values will be approximately the same. If a mode has this structure as well, they will have some resemblance with F and appear in the projection. In Figure 6.1, we can see that this is the case for the odd modes and not the case for the even modes.

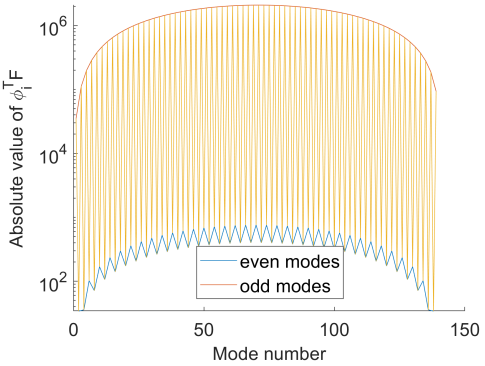


Figure 6.2: Values for $\Phi^\top F$. Modes that do not have resemblance with the right-hand side have a small $\phi^\top F$.

This can also be observed in Figure 6.2. Note that, since the right-hand side changes over time, the values in this figure will also change over time. In this example setup, the difference between the inner product with the odd modes and with the even modes remains substantial over time.

This problem appears because in Equation 5.5, we can see that the right-hand side takes no part in the computation of modes. As a consequence, relevant information in the right-hand side is not included in the modes. In this system, there is relevant information of the system in the right-hand side. Namely, the boundary conditions that are eliminated are included in the right-hand side. Therefore the modal analysis misses crucial information about the governing system. This results in modes that have do not include information on the forcing vector, but still have a low frequency. Modal decomposition recognizes these modes as high influence modes because of the low frequency. However these modes are actually of low influence, because when projecting onto these modes, information of the right-hand side is lost.

When running this simulation for a different number of modes with modal decomposition, we get the result in Figure 6.3. First, we treated the modes with the lowest frequency as most important. In the other simulations, we treated odd modes as more important than even modes, and within the odd and even modes the lowest frequency as most important.

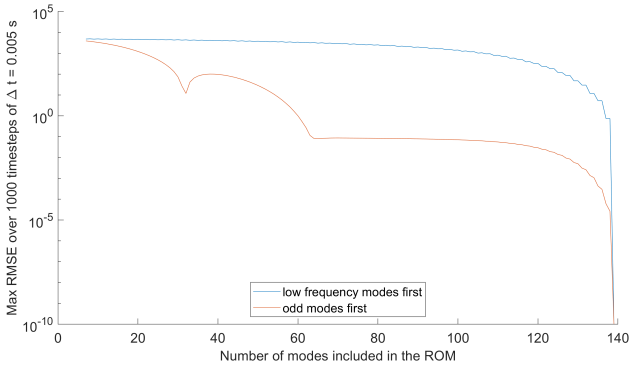


Figure 6.3: RMSE for a modal decomposition ROM for the configuration with 40 modes on one roller segment and 70 modes on one free-belt segment.

We can conclude for this example setup that treating the odd modes as more important modes in the projection gives better results. However, since the right-hand side in Equation 6.9 changes for

different boundary conditions on the rollers and changes over time, we cannot define a general way to include the most important modes for this situation. Next to that, the information in the right-hand side is not included in the modal analysis, so also in the modes itself (not the ordering only), crucial information can be missing.

For other time-invariant systems, this issue can arise because the forcing term is not involved in the computation of modes here as well. One solution can be to find a way of involving F in the computation of modes. Another potential solution involves reordering the importance of the modes based on the inner product with the right-hand side of the dynamic equations. For example, the ordering could be done based on the frequency divided by the inner product with the right-hand side. For POD, an approach where the modes are selected in a different order than the conventional ordering, is found in [43], where the selection of modes is modified by including both the frequency and the energy of modes, instead of only including the energy as is done in regular POD. However, since the approach we suggested requires the right-hand side to be known over time, it is not applicable to the system addressed in this thesis. It would be interesting to explore whether this method could be effective for systems where this issue occurs and the right-hand side is known in advance.

6.3. Proper Orthogonal Decomposition

In this section, we discuss the application of POD to the one-dimensional dynamical system. As previously stated, the algebraic constraints are not subject to reduction so will be eliminated from the system. This gives a full order model with a dimension of 139 as explained in section 6.2. For POD, the number of snapshots and reduction order have to be decided. In this section, we outline the necessary values and the considerations for their selection. In the next chapter, these considerations are taken into account for selecting the values for the two-dimensional system.

Snapshots

First of all, we have to choose the data which will be used in the POD analysis. We want to reduce only the dynamical system part by projecting it onto a smaller space in which the solution lives. Therefore we choose to use the data of solutions *before* interpolation ($\tilde{\mathbf{x}}_{i+1}$ in Equation 6.10). The solutions after interpolation ($\mathbf{x}_i$) are not a correct representation of the solution after the dynamical system step.

$$\mathbf{x}_i \xrightarrow{\text{Dynamical system}} \tilde{\mathbf{x}}_{i+1} \xrightarrow{\text{Interpolation}} \mathbf{x}_{i+1}. \tag{6.10}$$

There are two options on how to choose which simulation data is used for POD.

1. One choice for the simulation data that will be used in the POD, is to run part of the simulation and use those snapshots to predict the rest of the trajectory. This could lead to a better ROM, because we are sure that relevant and compatible data is used. However, for this, still a (shorter) full order simulation has to be done.

2. Another option is to use data of different simulations to calculate the POD modes. With this, we have to gather data from different simulations, which may take some time. But then, the same data will be used to reduce more than one simulation. As a result, after the data is gathered, we never have to run the full order model anymore.

Both options are investigated in this section.

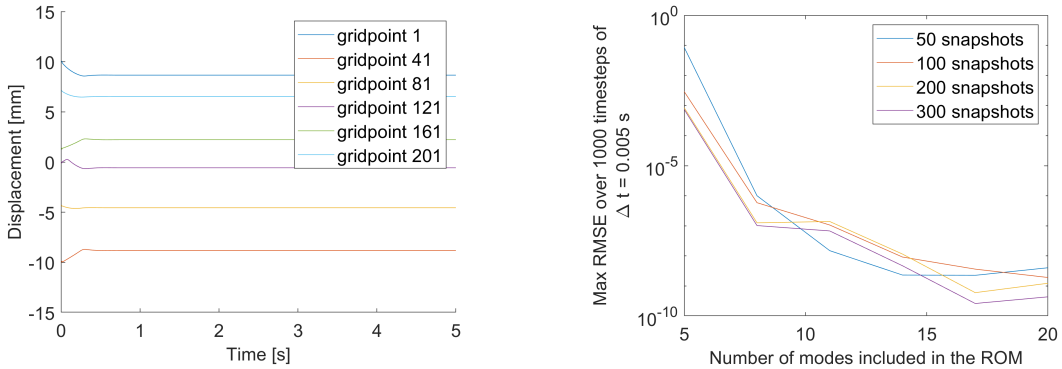
Dimension of the reduced order model

Another parameter that has to be chosen, is the dimension of the ROM. The choice for this dimension is a trade-off between the energy that is captured in the basis and the computational speedup. We will make this trade-off on the one-dimensional model in this section and this will be done in the same manner for the two-dimensional model in the next chapter.

6.3.1. Snapshot generation and POD: single configuration

First, POD is implemented with snapshots generated with the first part of the simulation. For long simulations, the steady state is most important. Therefore, it is expected that when the snapshots contain information on the steady state, the ROM would be accurate for long simulations. In Figure 6.4a,

the trajectory of a simulation with one roller that is translated by 10 mm is shown. After 0.5 seconds (100 timesteps), the displacement nearly has reached a constant value. Consequently, it is expected that using the solution prior to 0.5 seconds as snapshots will effectively capture the relevant behavior of the solution for a long simulation. In Figure 6.4b, we have shown the resulting RMSE for the same one-dimensional configuration for different number of snapshots. Note that this comparison is done for one specific setup of rollers and the belt, so the numbers are not directly applicable to other simulations. We can conclude for this simulation that taking 50 snapshots gives an appropriate error.

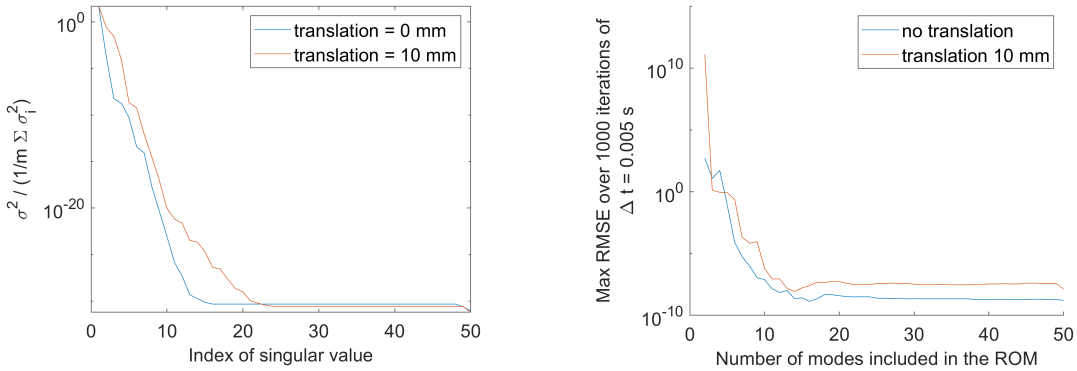


(a) Trajectory of displacement of tracer points in x direction for the first 5 seconds of the one-dimensional simulation with one translated roller.

(b) RMSE for a POD ROM using different number of snapshots and different number of modes.

Figure 6.4: Determination of number of modes for a configuration with one roller translated by 10 mm. The full order model has a dimension of 139.

With these snapshots, the reduced model is determined and executed for the rest of the simulation. The energy that a number of modes can capture is based on the singular values. For 50 snapshots, we have 50 different singular vectors in the POD analysis. The singular values and the resulting maximum error for a simulation of 1000 timesteps is shown in Figure 6.5 for two different configurations. In Figure 6.6, the computation time for different number of modes is shown. There seems to be a slight trend with slower computations for a higher number of modes, with some outliers, which is in accordance with the expectations. Note that the degrees of freedom is much lower than in a two-dimensional model. So also the size of the system is much lower and this may cause the overhead of the simulation to dominate in the computational time. The computational time required for the two-dimensional model will be evaluated more comprehensively.



(a) Relative singular values of the snapshot matrix based on two different configurations.

(b) RMSE for a POD reduced order model trained on a single configuration.

Figure 6.5: Results for a different number of modes in POD for the one-dimensional setup. In both cases, the full order model has a dimension of 139 and the reduced order model has a size equal to the number of modes included.

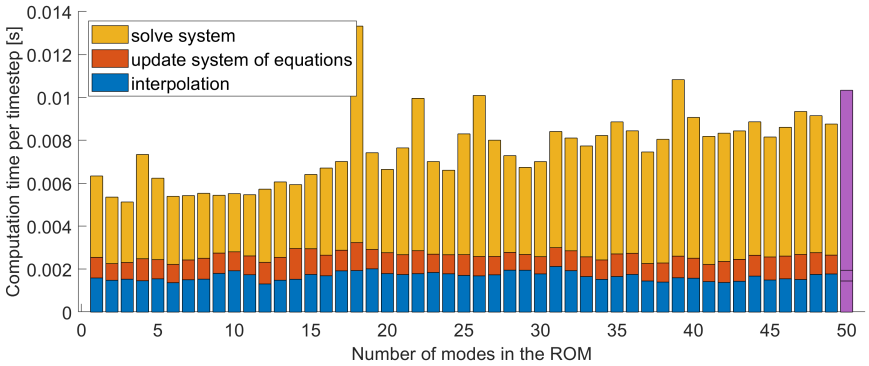


Figure 6.6: The mean computation time for the different code segements in one timestep. The number of modes correspond with the size of the reduced model. The purple boxes represent the full order model with a dimension of 139.

6.3.2. Snapshot generation and POD: multiple configurations

From the previous section, we could conclude that 50 snapshots are sufficient to capture the behaviour of one simulation. Therefore, for the approach to use different simulation data, the same number of snapshots for training the data is used. In the one-dimensional toy model, we do not have that many different configurations to choose from. Just the rotational speed and the translation of the rollers can be changed. In this example, training data listed in Table 6.1 is used. We want to test if we can simulate the system for other settings using POD trained on this data.

Table 6.1: Training data used in POD. Both training runs have been executed for 50 timesteps.

	Rotational velocity roller 1: z	Translation roller 1: x
Training run 1	10 rad/s	10 mm
Training run 2	7 rad/s	4 mm

In Figure 6.7, the RMSE for different runs is shown. These runs are all reduced by POD using the training data from Table 6.1. The reduced simulations for different setups give sufficiently small errors. Therefore, the expectation for the two-dimensional model is that POD trained on different simulation data can give a sufficient ROM.

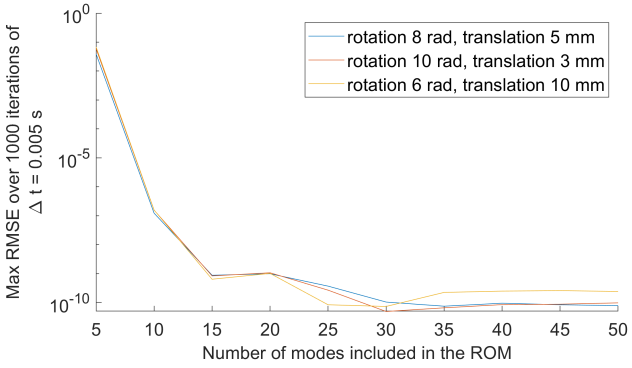


Figure 6.7: RMSE for a POD reduced order model trained on multiple configurations from Table 6.1. The full order simulations has a dimension of 139.

The advantage of training on multiple configurations is that the full order model does not need to be simulated again for generating snapshots. Instead, we will save some full order snapshot data, which can then be used for multiple other simulations with different configurations.

6.4. Dynamic Mode Decomposition

The advantage of Dynamic Mode Decomposition is, that it is not based on any of the governing equations, only on some simulation data. Therefore, with a satisfactory reduced model, the effect of the interpolation step is included in the DMD model and need not to be executed separately. In modal decomposition and POD, we reduced the eliminated system with only differential equations. In DMD however, because we want to keep the advantage of not using the governing equations, we do not want to calculate the eliminated variables from the dynamic variables, because this will be done using the governing equations. Therefore, the snapshots are build from both the algebraic and dynamic variables. Hence, for the one-dimensional model with two rollers that can have a translation and rotation, the state has the structure

$$\mathbf{x} = \begin{bmatrix} \mathbf{u}_{r_1} \\ \mathbf{u}_{f_1} \\ \mathbf{u}_{r_2} \\ \mathbf{u}_{f_2} \\ d_1 \\ \alpha_1 \\ d_2 \\ \alpha_2 \end{bmatrix} \in \mathbb{R}^N. \tag{6.11}$$

The model is based on 40 nodes on the roller segments and 70 nodes on the free-belt segment and therefore $N = 224$. DMD is designed for first order dynamical systems, so we have to transform the second order dynamical system to a first order dynamical system. Consequently, we use

$$\begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}$$

as snapshots in the DMD analysis. This gives a snapshots matrix of size $(2N \times m) = (448 \times m)$. where m is the number of snapshots. There are four things to decide when applying DMD on this system.

1. Taking snapshots before interpolation or after interpolation ($\tilde{\mathbf{x}}_{i+1}$ versus $\mathbf{x}_{i+1}$ in Equation 6.10).
2. Training with snapshots of the same single configuration or training with snapshots of multiple different configurations.
3. The number of snapshots.
4. The dimension of the reduced order model.
5. How to select the modes that are relevant in the solution.

Considering the first decision: the snapshots before interpolation contain more information on the system, specifically the mean displacement (that is subtracted in the interpolation step, so not observable after interpolating). Therefore it became apparent that better results are obtained when the state before interpolation is used as snapshots. Interpolation can then be seen as a post-processing step. That means, with DMD, we will approximate the operation from $\tilde{\mathbf{x}}_k$ to $\tilde{\mathbf{x}}_{k+1}$.

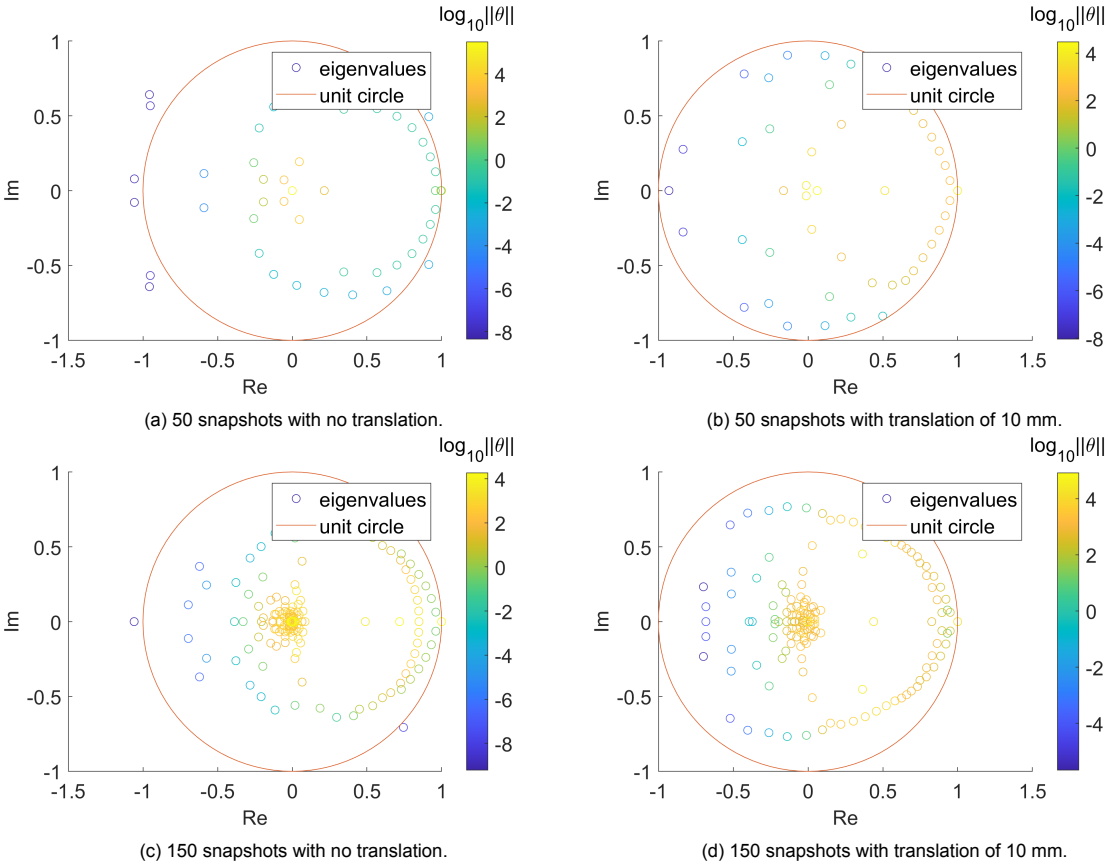
The other three decisions will be explained in the following two subsections. First, training with snapshots of the same single configuration is discussed and after that, training with multiple different configurations.

6.4.1. Snapshot generation and DMD: single configuration

When snapshots are taken from the same simulation, a linear model is created with DMD on these snapshots, which can be used for the rest of the simulation. For a different number of snapshots, the eigenvalues of the full matrix A are shown in Figure 6.8 with and without translation. The snapshots from the full order model have a dimension of 448, so the matrix A is of size 448×448 and has in total 448 eigenvalues, for each number of snapshots. The modes with an eigenvalues greater than 1 in absolute value can diverge, because the solution increases exponentially with the number of timesteps k, as $e^{\lambda k}$. Without translation and 50 snapshots, the are some diverging modes with a significant norm of their scaled mode θ. This gives the impression that these modes do not give a correct representation

of the solution. When using 150 snapshots, the modes with an eigenvalue greater than 1 in absolute value have a low norm. This gives a reason to classify them as insignificant and to ignore these modes. In the DMD analysis with translation, no diverging modes can be found for 50 and 150 snapshots as is visible in Figure 6.8.

It is decided that for the DMD analysis of these simulations, 150 snapshots will be used. This results in a ROM with a dimension of 1 to 150, depending on how much modes will be included in the solution.



(a) 50 snapshots with no translation.

(b) 50 snapshots with translation of 10 mm.

(c) 150 snapshots with no translation.

(d) 150 snapshots with translation of 10 mm.

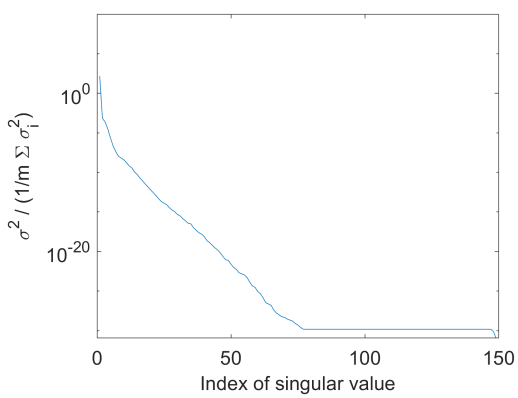
Figure 6.8: DMD eigenvalues in the complex plane. The color indicates the norm of toe associated scaled mode.

For the case with one roller translated by 10 mm, we look at the singular values and the error to determine how many modes should be taken into account. The singular values and RMSE for a dataset of 150 snapshots are shown in Figure 6.9. The number of modes that are included in the analysis is based on the amplitude of the corresponding eigenvalues. In the DMD analysis, we exclude the DMD modes that have an eigenvalue with an amplitude higher than 1 in the solution state.

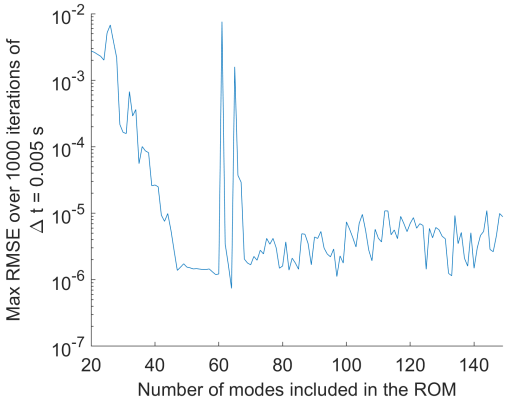
To summarize, the solution over time can be approximated by DMD. The one-dimensional model of dimension 224 will be rewritten into a first order dynamical system with 448 unknowns. For generating a ROM with DMD, 150 snapshots are used. The number of modes needed for simulating an accurate reduced model is roughly 20. Note that, the two-dimensional model has some differences with this one-dimensional model, so this is not generalizable to the two-dimensional setup. However, we will use a similar approach to select the number of snapshots, the dimension of the reduced order model and the selection of relevant modes.

6.4.2. Snapshot generation and DMD: multiple configurations

Like in POD with multiple configurations (subsection 6.3.2), we will also consider DMD that is trained on multiple different configurations. With DMD, the snapshot matrix is based on one trajectory, but this can be extended to multiple trajectories [40]. When we use q different trajectories, the snapshot



(a) Relative singular values of the snapshot matrix.



(b) RMSE for a DMD reduced order model trained on a single configuration.

Figure 6.9: Results for DMD trained on a single configuration where one roller is translated with 10 mm.

matrices look like

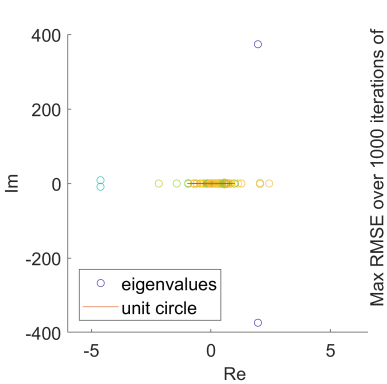
$$X = [\mathbf{x}_1^1, ..., \mathbf{x}_m^1 \quad , \mathbf{x}_1^2, ..., \mathbf{x}_m^2 \quad , ..., \mathbf{x}_1^q, ..., \mathbf{x}_m^q \quad] \in \mathbb{R}^{N \times qm}, \tag{6.12}$$

$$X' = [\mathbf{x}_2^1, ..., \mathbf{x}_{m+1}^1, \mathbf{x}_2^2, ..., \mathbf{x}_{m+1}^2, ..., \mathbf{x}_2^q, ..., \mathbf{x}_{m+1}^q] \in \mathbb{R}^{N \times qm}. \tag{6.13}$$

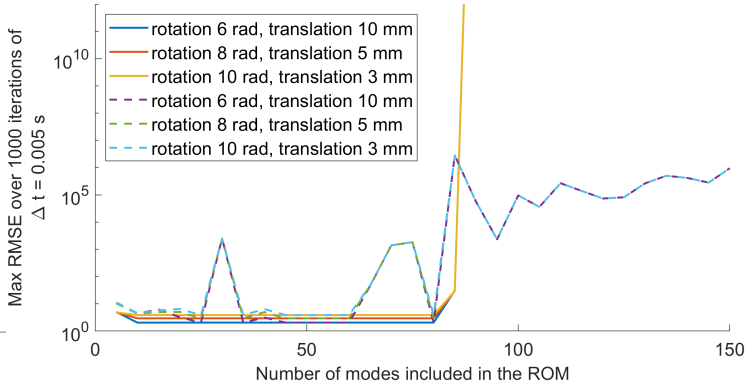
This is possible because the operator A relates the columns of X to those of X' in a pairwise fashion.

In this way, one set of simulation snapshots can be used for reducing a number of different new configurations. Once the simulation data is generated, no full order simulation has to be done for reducing a simulation with new configurations. So the initial value determines the trajectory of one simulation. For testing this approach on the one-dimensional system, the same setup is used as in subsection 6.3.2, so with snapshots of size $2N = 448$. Because DMD appears to need more snapshots than POD when training on data from the same configuration, we use more snapshots here as well. DMD is trained on 150 snapshots with both setups from Table 6.1.

The eigenvalues of the operator A are depicted in Figure 6.10a. While some eigenvalues are located within the unit circle, a considerable number has an amplitude exceeding 1. The norm of these scaled modes, represented by the color in Figure 6.10a, is substantial. This observation leads to the conjecture that the operator A does not accurately approximate the system. Even when the unstable modes are excluded from the solution, the result remains inaccurate. The resulting RMSE, both with and without the exclusion of unstable modes, is illustrated in Figure 6.10b.



(a) DMD eigenvalues in the complex plane. The color indicates the norm of the associated scaled mode.



(b) RMSE for a DMD reduced order model trained on multiple configurations. The dashed lines indicate the RMSE when the unstable modes are ignored in the solution.

Figure 6.10: Results for DMD trained on multiple configurations.

It can be concluded that DMD does not give appropriate results when training on simulations with different misalignments. This may be due to the fitted operator A accounting for the specific misalignments, which differ in the test simulations. One potential solution is to treat the misalignments as control

variables. The method introduced by Proctor et al. [44], known as Dynamic Mode Decomposition with control (DMDc), has the capability to distinguish between the underlying dynamics and the effects of actuation, leading to an accurate reduced DMD model. Due to time constraints, this thesis will not investigate or implement DMDc. Future research could explore this approach to potentially accelerate the conveyor belt model.

7

Numerical results for the two-dimensional model

In the previous chapter, we demonstrated that modal decomposition is unsuitable for generating an accurate reduced order model for this application. POD was shown to produce an accurate reduced-order model when an appropriate set of snapshots was selected. POD did not facilitate the reduction of the interpolation step. Conversely, DMD allowed for the inclusion of the interpolation step in the reduction process, yielding an accurate reduced order model when trained on snapshots from simulations with identical configurations. However, when DMD was trained on snapshots from simulations with different configurations, the resulting DMD modes failed to produce an accurate reduced order model.

Given the unsuitability of modal decomposition for this application, and considering that the argumentation in the previous chapter extends to the two-dimensional model, we opted not to implement modal decomposition on the two-dimensional model. We will apply POD and DMD to the two-dimensional model using the same methodologies described in the previous chapter. The results will be evaluated in this chapter.

Firstly, the performance measures used for the one-dimensional model are extended to the two-dimensional model in section 7.1. Here, the code segments that contribute the most to the total computational time are explained, as they are defined differently compared to the one-dimensional model. The results in this chapter are based on one general setup, detailed in section 7.2. For other setups, the structure of the equations remains consistent, with dynamic equations on the free-belt segments and algebraic equations on the roller segments. Consequently, similar results are expected, and other setups are not considered in this thesis. In section 7.3, we present the details and results for the two approaches of POD. In section 7.4, we present the results for DMD, briefly mentioning the approach of DMD based on data from simulations with different configurations to show that it yields the same inaccurate result as for the one-dimensional setup. In section 7.5, we compare the results of POD and DMD.

There are some key differences between the one-dimensional and two-dimensional models that need to be addressed. In a two-dimensional conveyor belt system, there are more degrees of freedom for the belt and the motion of the rollers, resulting in a larger full order model. We cannot directly transfer the parameters from the previous section, such as the number of snapshots and the number of modes, and these will be reanalyzed in this chapter. Additionally, the number of cases with different misalignments is significantly higher, resulting in more misalignment combinations to investigate.

7.1. Performance measures

In the previous chapter, the Root Mean Square Error and Relative Root Mean Square Error were introduced and explained in section 6.1. These metrics will also be used in this chapter. To evaluate the performance of a method without simulating the full order model, error bounds or error estimations can be calculated as suggested in various studies [45], [46], [47], [48]. However, since these estimators involve computations with the governing equations of the entire model (this would mean including the

interpolation step) or the development and training of new machine learning models, they are beyond the scope of this thesis.

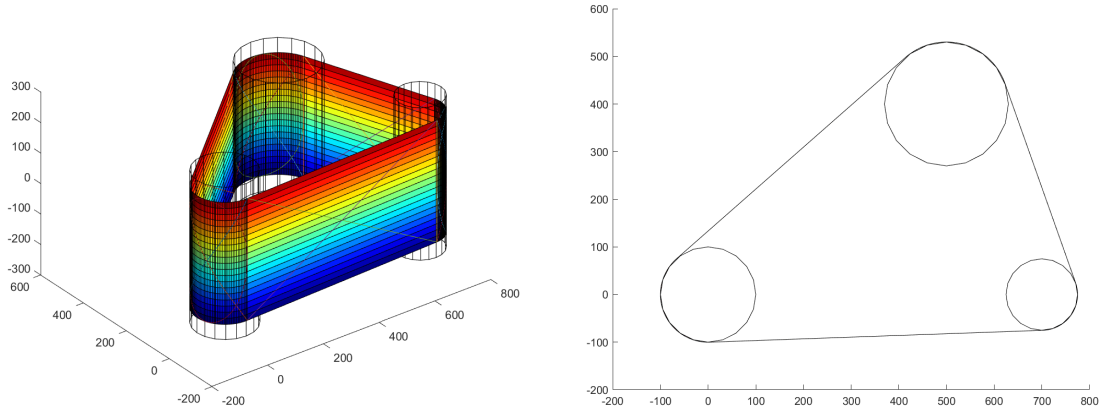
In addition to the error metrics, the computational time of the most dominant code segments is evaluated. For the two-dimensional setup, the following functions are the most significant in terms of computational time:

- **SolveDynamicSystem**: Similar to 'solve system' in the one-dimensional case, this function solves the system of dynamic and algebraic equations.
- **TransportDisplacements**: Similar to 'interpolation' in the one-dimensional case, this function interpolates the solution after solving the system.
- **CreateEquationMatrix**: Similar to 'update system equations' in the one-dimensional case, this function creates the system of equations.
- **SubstituteODE**: This additional function is executed when algebraic variables are eliminated as described in subsection 3.4.3. In this function involves both the substitution of algebraic variables into the system of dynamic equations before solving, and the calculation of the algebraic variables from the dynamic variables after solving.

The measured timing values are the mean of the computational time of the function over all timesteps. To measure the computational time, five consecutive simulations were conducted on a computer with a 12th Gen Intel(R) Core(TM) i7-12700 with 12 cores, 20 threads, 2.1 GHz, and 16GB of internal memory using Matlab R2024a.

7.2. Test setup

We use a setup with three rollers, as illustrated in Figure 7.1. This setup is chosen because a system with three rollers is expected to exhibit more complex behavior than a system with two rollers. Although the simulation program can be used for systems with up to six rollers, adding more than three rollers is not expected to significantly increase complexity. In this test setup, the three rollers are positioned upright, sharing the same central z position and length, but differing in radius. Roller 1, located at position (0,0), functions as the drive roller such that the system has a belt velocity of 1 m/s. The total duration of the simulation is 100 seconds.



(a) Side view.

(b) Top view.

Figure 7.1: Test setup used in this chapter. Axes are given in millimeters.

The grid size is variable, allowing us to test the computational time for different full order model sizes. After setting the grid size, tracer points are placed at specific nodes on the belt. During the simulation, the displacement of these tracer points is recorded. The RMSE and RRMSE are computed based on the tracer points only. The placement of tracer points can be customized according to user preferences. In this thesis, five tracer points are evenly distributed per belt segment for all simulations.

In the one-dimensional toy model, misalignment could only be introduced through translation in a single direction. In this two-dimensional system, misalignments can be introduced through rotations in two directions (excluding the moving direction) and translations in three directions.

7.2.1. Computational complexity and creating a baseline

The two-dimensional full order model can be simulated with the approaches explained in section 3.4. The most efficient approach will be used as baseline when analyzing the results generated by the reduced order models. The computational complexity of the different approaches is summarized in Table 7.1. This complexity is based on the expected number of flops (floating-point operations). The number of unknowns N is generally in the order of a few thousands. The bandwidth b for this system is in the order of the number of grid elements in the z-direction n_η, because the indices of two connected nodes in the x direction are separated by $2n_\eta$. The number of algebraic equations is approximated to be $1/3N$, since approximately $1/3^{\text{rd}}$ of the belt is on a roller segment.

Table 7.1: Computational complexity of different approaches for a system with N unknowns. The number of unknowns for the algebraic equations is indicated with N_a and the bandwidth of the stiffness matrix is indicated with b.

Code segment	Naive approach	Elimination of variables	Direct time integration
TransportDisplacements	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
CreateEquationMatrix	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
SubstituteODE	-	$\mathcal{O}(N_a)$	-
SolveDynamicSystem	$\mathcal{O}(3Nb^2)$	$\mathcal{O}((N-N_a)b^2)$	$\mathcal{O}(Nb^2)$

In the function **TransportDisplacements**, the interpolation step is performed. For each grid point, the displacement in the x direction must be interpolated, and the mass must be transported. Since this function is treated as a black-box, we estimate its complexity to be of order N (since N is approximately twice the number of grid points), because for each node, a small number of flops have to be done.

In the function **CreateEquationMatrix**, the matrix used in the naive approach is set up. Each row is filled with one equation involving a small number of unknowns. Investigating and optimizing this function is beyond the scope of this thesis; thus, the computational complexity to compute a small number of values (in the order of 1) for all N rows is estimated to be of order N.

The function **SubstituteODE** is used only in the elimination of variables approach. To determine the computational complexity, we define N_a to be the number of variables that are determined by the algebraic equations. For each of these variables, the corresponding row must be subtracted from all the rows where this variable contributes to. Both the number of rows where this variable contributes to, and the number of nonzeros in these rows are assumed to be in the order of 1. Therefore, this operation approximately uses N_a flops.

The function **SolveDynamicSystem** solves the dynamical system. In the naive approach, we solve a linear system of $3N$ unknowns with a bandwidth b. This has a time complexity of $3Nb^2$. This is reduced to Nb^2 when direct time integration with Backward Euler is used. The Backward Euler scheme is shown in Equation 7.1. The first step is solving a sparse system of size N with a bandwidth b, which has a time complexity of order Nb^2.

$$\begin{aligned} \text{Solve } (M+\Delta t^2 K)\ddot{\mathbf{x}}_k &= F - K(\mathbf{x}_{k-1} + \Delta t \cdot \dot{\mathbf{x}}_{k-1}) \text{ for } \ddot{\mathbf{x}}_k, \\ \dot{\mathbf{x}}_k &= \dot{\mathbf{x}}_{k-1} + \Delta t \cdot \ddot{\mathbf{x}}_k, \\ \mathbf{x}_k &= \mathbf{x}_{k-1} + \Delta t \cdot \dot{\mathbf{x}}_k. \end{aligned} \tag{7.1}$$

To establish a baseline, we compare the naive approach and direct time integration across different grid sizes. The elimination of variables is not considered here, as it appeared to introduce additional computational time due to the separation of the time integration into the SubstituteODE and SolveDynamicSystem functions, making it less efficient than direct time integration in terms of computational time. For smaller grid sizes, the upper bound for the timestep is more restrictive due to the CFL condition for the interpolation step, resulting in an increased number of timesteps required to simulate a fixed time interval. We ensure that there are at least five grid elements in the moving direction on the smallest roller. The other element sizes are adjusted so that all elements are approximately the same size and close to square. The total number of unknowns on the elements is denoted by n. Figure 7.2 illustrates the computational time for solving the dynamical system. These simulations do not include misalignments, as the primary focus here is to evaluate computational time.

From Table 7.1, we observe that, theoretically, the computational time can be reduced by a factor of 3 when using direct time integration instead of the naive approach. The difference shown in Figure 7.2

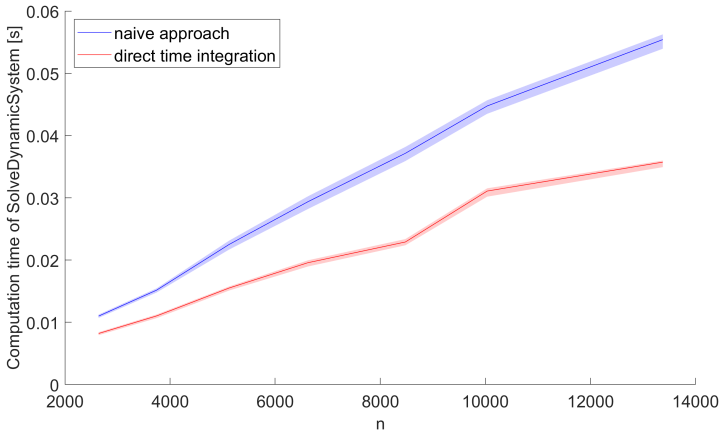


Figure 7.2: Computational time gain for different mesh sizes by reformulation of the system. The shaded area indicates the maximum and minimum over the timings.

between the two approaches does not reflect this factor of 3. However, the reduction of computational time increases with the number of grid elements. This may be due to additional operations required in the reformulation of the system when using direct time integration, which have less impact on the total computational time as the number of elements increases.

Based on this analysis, direct time integration will be used to simulate the full order model in the rest of this chapter.

7.3. Proper Orthogonal Decomposition

In this section, POD will be applied by training on the same single configuration and by training on multiple configurations. In both cases, the number of snapshots and number of modes needed for an accurate reduced order model have to be determined. Additionally, for the reduced model trained on different simulations, we have to specify how to choose the training data.

7.3.1. Snapshot generation and POD: single configuration

The implementation is analogous to POD with snapshots from the same simulation in the one-dimensional case (see subsection 6.3.1). First, we simulate a a number of timesteps of $\Delta t = 0.02$ seconds with the full order model. The snapshots generated in those timesteps will be used for the POD analysis. The resulting ROM will be used to simulate the rest of the simulation. A grid of 18×104, corresponding to 3762 unknowns, is used with the following misalignment configuration.

Table 7.2: Settings for POD using snapshots with the same configuration.

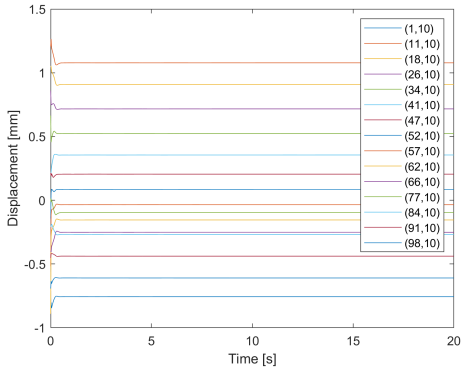
	Translation roller 1: y^{3D}	Rotation roller 2: x^{3D}	Translation roller 3: x^{3D}
Test run 1	3 mm	4e-4 rad	-1 mm

Number of snapshots

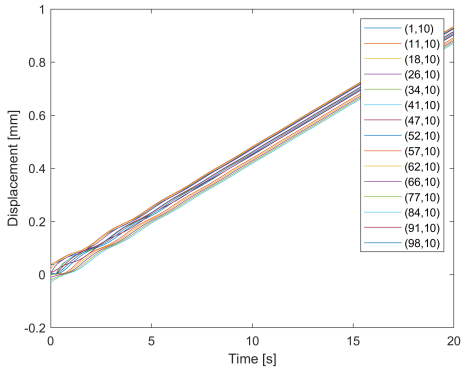
The trajectory of the x and z displacement of this simulation for the first 20 seconds is shown in Figure 7.3. When running long simulations, the belt will approach a state in which the belt maintains a constant velocity. This state is referred to as the 'steady state' in this context. For a ROM that can predict a long simulation accurately, it is important that the snapshots contain information on the steady state. By visually observing the trajectory of the start of the simulation, we take the trajectory of the first 8 seconds, corresponding to 400 snapshots of $\Delta t = 0.02$, as snapshot data. In this way, the snapshots should contain information on the steady state.

Number of modes

With 400 snapshots, the snapshot matrix is created and a singular value decomposition is computed. The singular values of the snapshots matrix are shown in Figure 7.4a. Based on the rapid decrease



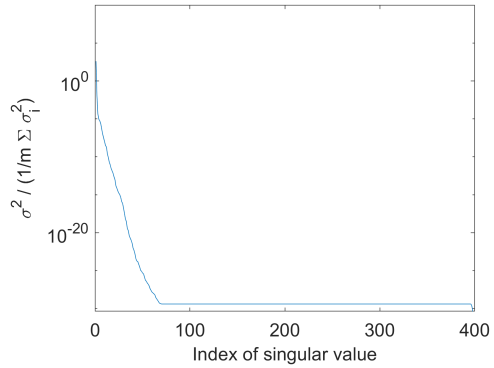
(a) Trajectory of the (interpolated) x displacement.



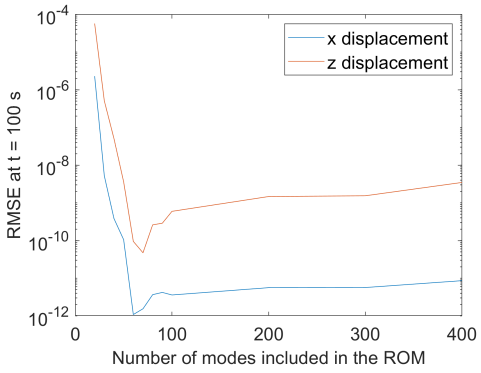
(b) Trajectory of the z displacement.

Figure 7.3: Trajectory of displacement of tracer points in x and z direction for the first 20 seconds of one simulation with the configuration in Table 7.2.

for the first singular values, we can expect not to need a lot of modes for an appropriate reduced model. After around 70 modes, the singular values are close to zero and not decreasing much anymore. Therefore we can expect that the error does not decrease much after 70 modes as well.



(a) Relative singular values of the snapshot matrix based on a single configuration.



(b) RMSE for a POD reduced order model at 100 seconds.

Figure 7.4: Results for a different number of modes in POD for the two-dimensional setup using 400 timesteps as snapshot data. The settings are listed in Table 7.2. In both cases, the full order model has a dimension of 18×104 and the reduced order model has a size equal to the number of modes included.

The RMSE of simulations with varying numbers of modes is shown in Figure 7.4b. Including too many modes can reduce the impact of the most important modes in the reduced order model, which could explain the fact that higher errors occur with an increased number of modes after using more than 60 modes. For the conveyor belt systems studied in this thesis, the precision tolerance is 20 µm. Since the RMSE is in mm, a RMSE of 0.02 mm meets this tolerance requirement. When including 20 modes or more, the RMSE is below 0.02 mm. The RMSE over the total simulation when using 400 snapshots and 20 modes is shown in Figure 7.5. The mean computational time per timestep for solving the system (SolveDynamicSystem) and substituting the algebraic variables (SubstituteODE) after generation of snapshots is shown in Figure 7.6.

The two figures demonstrate that using POD it is possible to obtain a reduced order model that can accurately replicate the full order model while reducing simulation time when training on snapshots with the same configuration.

The number of snapshots required for an accurate reduced order model should be determined by analyzing the initial phase of the simulation. For simulations with a similar setup and drive roller velocity, we expect the number of snapshots for an accurate reduced order model to be approximately the same. This is because the belt will be in a comparable state after a fixed period of time for different misalignments with the same belt velocity.

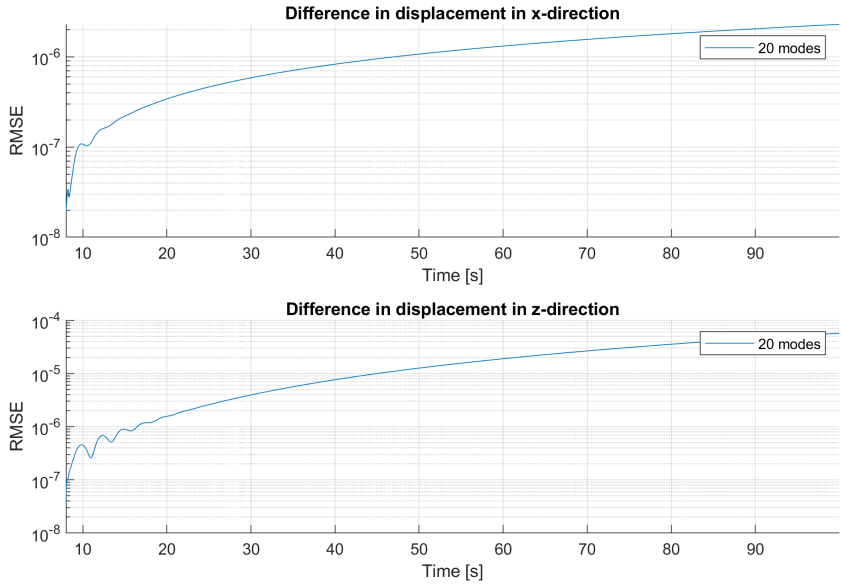


Figure 7.5: RMSE for a POD reduced order model trained on a single configuration (Table 7.2) using 400 snapshots.

The number of modes to include in the reduced order model will be based on the singular value decomposition of the snapshot matrix. To assure that the error is well below the tolerance and we still obtain a speedup, we can conclude that modes where

$$\frac{\sigma^2}{\frac{1}{m}\sum_{i=1}^{m}\sigma_i^2} < 10^{-20} \tag{7.2}$$

will be included in the reduced order model.

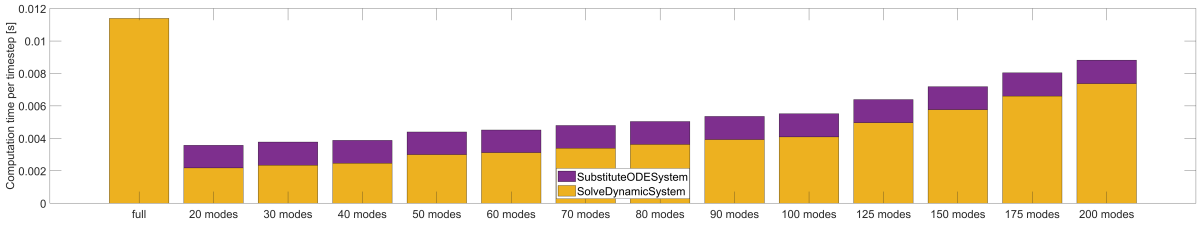


Figure 7.6: The mean computation time for solving the system in one timestep in POD reduced order models with different number of modes.

To evaluate the scalability of POD, the method was applied to a larger system of equations. The singular values for these systems, along with those for the reference system of 18×104, are presented in Figure 7.7. This indicates that a similar number of modes is required to achieve an accurate reduced order model. The RMSE for a system with a grid of 21×122, resulting in 2580 unknowns, shows this as well, shown in Figure 7.8. This demonstrates that even for systems with a higher number of grid points, the dimension of the reduced order model remains comparable, thereby confirming the scalability of POD.

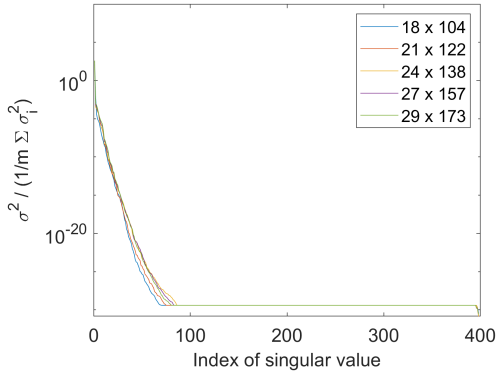


Figure 7.7: Relative singular values of the snapshot matrix based on a single configuration for two different system sizes.

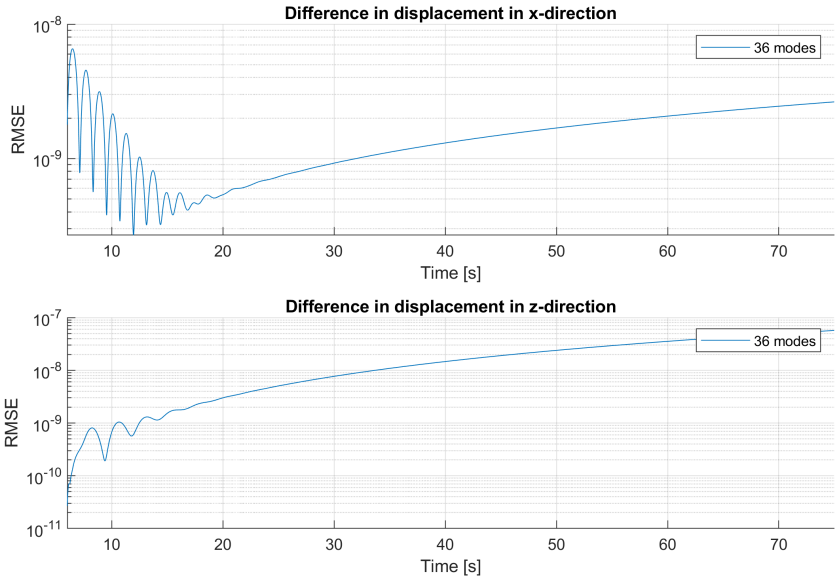


Figure 7.8: RMSE for a POD reduced order model trained on a single configuration (Table 7.2) with a grid of 21×122 using 400 snapshots.

7.3.2. Snapshot generation and POD: multiple configurations

In this section, the POD reduced order model will be constructed using snapshots from multiple configurations. The same setup with a grid of 18×104 as in the previous section is used. The key difference from POD trained on a single configuration lies in the selection of the training data. Once the training data is chosen, the next step is to determine the number of modes to include in the reduced order model.

Training data

In section 6.3, we have trained with snapshots from two simulations. These simulations contained misalignments in only one direction. In two-dimensional conveyor belt systems, we can vary rotations in two directions (not the rotation in moving direction) and translations in three directions [1]. This increases the amount of possibly useful training data. Therefore, it is important to have a clear method for defining the training data. The first rule we establish is that the testing data must include the same misalignments as the training data, only possibly with different values. This means that translations or rotations in directions not present in the training data should not be used in the testing data.

In Table 7.3, an initial example of training and testing data is shown. The timestep and the rotational velocity of the drive roller are the same for all the runs. Because of the rule mentioned in the previous paragraph, this training data cannot be used for a simulation where a translation in y^{3D} direction is

[1] The rotation in moving direction is not varied since the belt always has a speed of around 1 m/s in the applications considered in this thesis.

applied on the third roller for example. We perform POD on simulations that vary in how closely they match the training data. These simulations of 10 seconds, help us to evaluate the effectiveness of the POD trained on multiple different configurations. After analysing these results, we will formulate a general way to determine the correct training data.

Table 7.3: Training data and test data used in POD. Both training runs have been executed for 50 timesteps.

	Translation roller 1: y^{3D}	Rotation roller 2: x^{3D}	Translation roller 3: x^{3D}
Training run 1	1 mm	5e-4 rad	-1 mm
Training run 2	2 mm	2e-4 rad	1 mm
Test run 1	3 mm	4e-4 rad	-1 mm
Test run 2	1 mm	1e-3 rad	2 mm
Test run 3	2 mm	5e-4 rad	-0.5 mm
Test run 4	1 mm	5e-4 rad	-1 mm
Test run 5	2 mm	2e-4 rad	1 mm
Test run 6	1.5 mm	3e-4 rad	0.5 mm

The resulting RRMSE for these runs is shown in Figure 7.9. For this initial example, the POD reduced order model contains 100 modes. The number of modes will be reduced later in this section. The actual displacement will differ for all test runs. Therefore, to make a fair comparison of the accuracy, the RRMSE is compared here instead of the RMSE.

For test runs 1 and 2, the RRMSE in both directions is relatively high compared to the other test runs. This may be because, in both of these test runs, one of the misalignment values is outside the range set by the training runs. It is noteworthy that test run 4 also has a relatively high RRMSE in the x direction, despite being identical to training run 1. This could be due to a too low number of snapshots in the training data. Additionally, the values continue to increase for most test runs, which could introduce problems when the simulation time exceeds 10 seconds. However, this issue might be resolved as well by taking a higher number of snapshots.

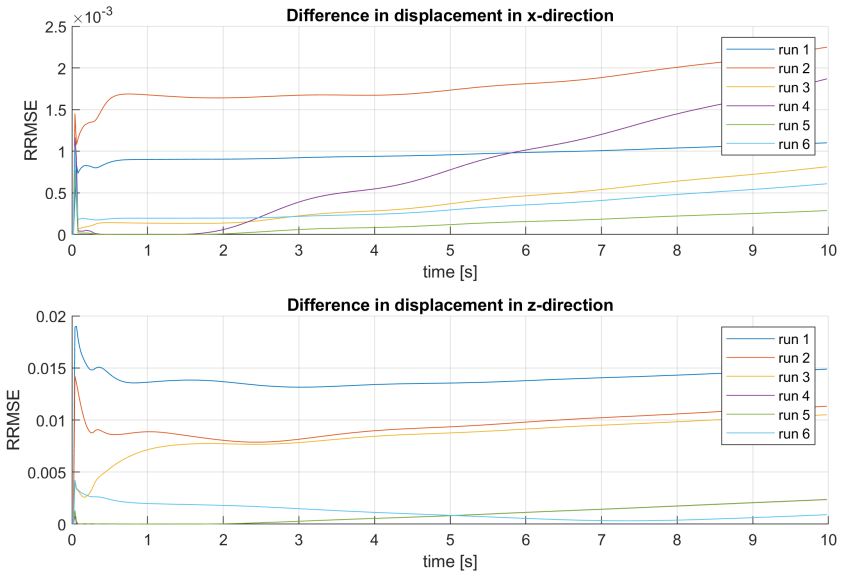


Figure 7.9: RRMSE for a POD reduced order model trained and tested on multiple configurations from Table 7.3.

Based on these initial observations, we will continue to investigate POD trained on multiple simulations with a more systematic approach. In this systematic approach, we aim to avoid any randomness in choosing the training set. First, the maximum rotational and translational misalignment are established, which are 0.5 milliradians (mrad, where 1 mrad $= 10^{-3}$ rad) and 5 mm, respectively. For a comprehensive study, two different misalignments are examined on each roller. For the first roller, we

consider combinations of translations in the x^{3D} and y^{3D} directions. These translations do not interact; this means that when a translation in y^{3D} is applied, the translation in x^{3D} remains constant for every grid point. For the second roller, we apply combinations of rotations in the x^{3D} and y^{3D} directions. Here, the interactions are more complex: when a rotation in x^{3D} is applied, both the y^{3D} and z^{3D} coordinates of the grid points change. When the rotation in y^{3D} is applied, the x^{3D} coordinates change, and the z^{3D} coordinates change again. On the third roller, a rotation in x^{3D} and a translation in y^{3D} direction are applied, which do also interact. The training data is generated by setting each misalignment to the maximum value and the other misalignments to zero. With this, six training runs are executed for 50 timesteps, listed in Table 7.4.

Table 7.4: Systematic set of training data used for POD. All training runs have been executed for 50 timesteps.

	Translation roller 1: x^{3D}	Translation roller 1: y^{3D}	Rotation roller 2: x^{3D}	Rotation roller 2: y^{3D}	Rotation roller 3: x^{3D}	Translation roller 3: y^{3D}
Training run 1	5 mm	-	-	-	-	-
Training run 2	-	5 mm	-	-	-	-
Training run 3	-	-	0.5 mrad	-	-	-
Training run 4	-	-	-	0.5 mrad	-	-
Training run 5	-	-	-	-	0.5 mrad	-
Training run 6	-	-	-	-	-	5 mm

Number of modes

Due to an implementation choice, the first three timesteps of a simulation correspond to an initialization process and will not be used as snapshots. As a result, this training data set contains a total of 282 snapshots. The SVD of this snapshot matrix yields the singular values shown in Figure 7.10. The same criterium for selecting the number of modes is applied here as in the previous method: modes with a singular value that satisfies

$$\frac{\sigma^2}{\frac{1}{m}\sum_{i=1}^{m}\sigma_i^2} < 10^{-20} \tag{7.3}$$

are included in the reduced order model. In this example, this results in 108 modes being selected.

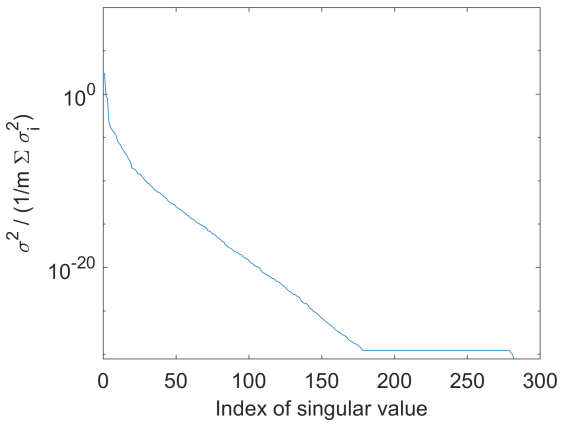


Figure 7.10: Relative singular values of the snapshot matrix based on configurations from Table 7.4.

Different test runs are simulated with POD trained on this data. The full list of test runs is summarized in Appendix A. The full order model has dimension of order 3762 and the reduced order model a dimension of 108. The results of three test runs will be discussed. The configuration of these test runs is shown in Table 7.5. The resulting RMSE for these test runs is plotted in Figure 7.11. The resulting RRMSE for the rest of the test runs can be found in Appendix A. Run 4 and 5 both have a relatively high error compared to all the test runs, but the RMSE is still below the tolerance of 0.02 mm. Run

Table 7.5: Testing runs used for POD.

	Translation roller 1: x^{3D}	Translation roller 1: y^{3D}	Rotation roller 2: x^{3D}	Rotation roller 2: y^{3D}	Rotation roller 3: x^{3D}	Translation roller 3: y^{3D}
Test run 4	12.5 mm	-	-	-	-	-
Test run 5	-	-	-	-	-0.55 mrad	-
Test run 11	2.5 mm	2.5 mm	2.5 mrad	0.25 mrad	0.25 mrad	2.5 mm

11 includes an extreme outlier of the range defined by the training data: the rotation of roller 2 is five times as high as the maximum in the training data. Even with this configuration, the POD basis seems to approximate the solution space accurately. The resulting displacement on the tracer points in the last ten seconds of the simulation with the full order model and the reduced order model is shown in Figure 7.12. There is only a small visual difference between the z-displacements generated by the full order model and the reduced order model.

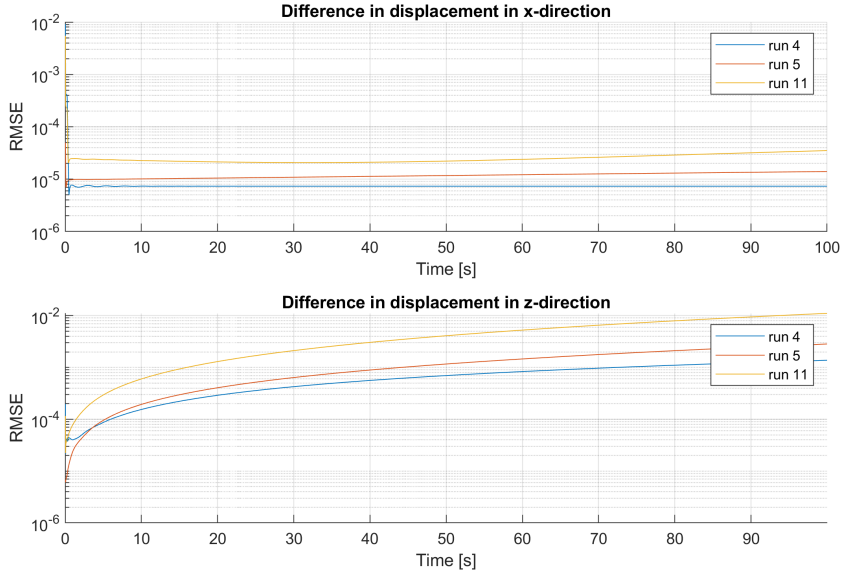


Figure 7.11: RMSE for a POD reduced order model with configurations from Table 7.5 trained on data with multiple configurations from Table 7.4.

The computational time of running the reduced order model is equivalent to that required for performing POD training on a single configuration for the same number of modes, as the same reduction method is applied to the full order model after mode selection. Therefore, the running time of the reduced order model is depicted in Figure 7.6. In this case, 108 modes are selected, which results in a computational speedup of approximately 2. This is lower than the speedup observed when executing one reduced order model timestep trained with the same configuration, since this requires less modes. However, this method eliminates the need to rerun full order simulations for reducing systems with new configurations, when the available training data already suffices.

7.3.3. Computational complexity POD

When considering the two options within POD—training on a single configuration versus training on multiple configurations-the computational complexity of running the reduced order model is equal. The differences of running the total simulation lie in the complexity of gathering the training data. The POD analysis, which has a very low computational cost, needs to be executed once. Therefore, we focus solely on the complexity of running a single timestep of the resulting reduced order model. The computational complexity of one reduced order model timestep with POD is listed in Table 7.6. In comparison to the complexity of the full order model, two key aspects become evident. First, POD is unable to reduce the functions TransportDisplacements and CreateEquationMatrix. Next to that, we observe that the reduced order model results in a dense system of equations. Consequently, even a

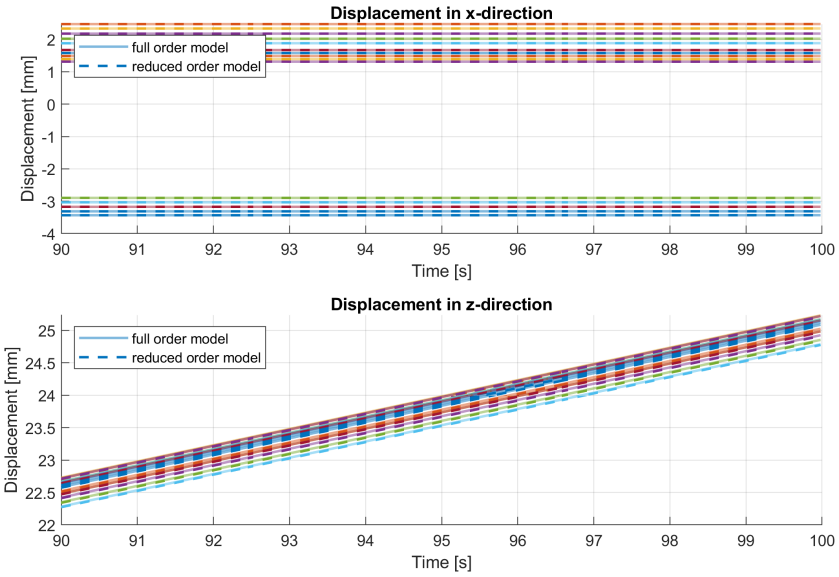


Figure 7.12: The trajectory of displacements of tracer points with the full order model (solid line) and the POD reduced order model (dashed line) for configurations from run 11 in Table 7.5 trained on data from Table 7.4.

small increase in the number of modes can quickly lead to longer computational times.

Table 7.6: Computational complexity of the POD ROM for a full system with N unknowns. The number of unknowns for the algebraic equations is indicated with N_a and the bandwidth of the stiffness matrix is indicated with b. The number of modes included in the POD ROM is indicated with r.

Code segment	Proper Orthogonal Decomposition
TransportDisplacements	$\mathcal{O}(N)$
CreateEquationMatrix	$\mathcal{O}\left(N\right)$
SubstituteODE	$\mathcal{O}\left(N_a\right)$
SolveDynamicSystem	$\mathcal{O}(r^3)$

7.4. Dynamic Mode Decomposition

In Dynamic Mode Decomposition, we observed that training on different simulation data for the one-dimensional conveyor belt system resulted in substantial errors. This observation also holds true for the two-dimensional belt simulation. As shown in Figure 7.13, the error in the z direction at the end of a 4 second simulation is depicted. This error is a result from training with the simulation data from Table 7.3 (a total of 100 snapshots) and testing it with the first test run. The resulting displacements are entirely different. We tested it for different configurations and settings as well and based on this result, we can conclude that training DMD with incorrect data yields insufficient results. Therefore, in our DMD analysis, we will focus on a reduced order model using snapshots with the same single configuration.

7.4.1. Snapshot generation and DMD: single configuration

To test DMD using snapshots from the same simulation, the same settings as for POD trained on a single configuration are used, shown in Table 7.2. We again use 18×104 grid elements that correspond to 3762 unknowns.

Number of snapshots

First, the number of snapshots required for an accurate approximation of the dynamical system is determined. The maximum number of modes are used here and the number of modes will be reduced later in this section. For DMD, it is crucial that the snapshots capture information on the steady state, as the DMD ROM relies solely on these snapshots and not on the governing equations. In Figure 7.14, the RMSE is shown for different number of snapshots.

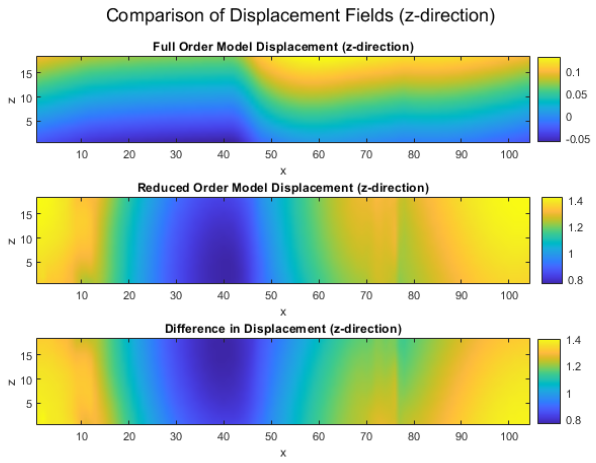


Figure 7.13: Displacements after 4 seconds for the FOM and the DMD ROM trained on multiple configurations.

It can be observed that using fewer snapshots can increase accuracy. One reason for this can be that in this figure, all modes are included in the reduced order model. When using 400 snapshots, some modes may not be relevant and could potentially increase the error. Despite that, we do not want to determine the number of snapshots based on the errors from different simulations, as this would require executing those simulations. Instead, we base the number of snapshots on the trajectory of the initial part of the simulation to maintain generality, and thus, 400 snapshots will be used in this analysis. For the same reason as with POD, 400 snapshots is sufficient to construct an accurate ROM, because this number of snapshots captures the steady state characteristics.

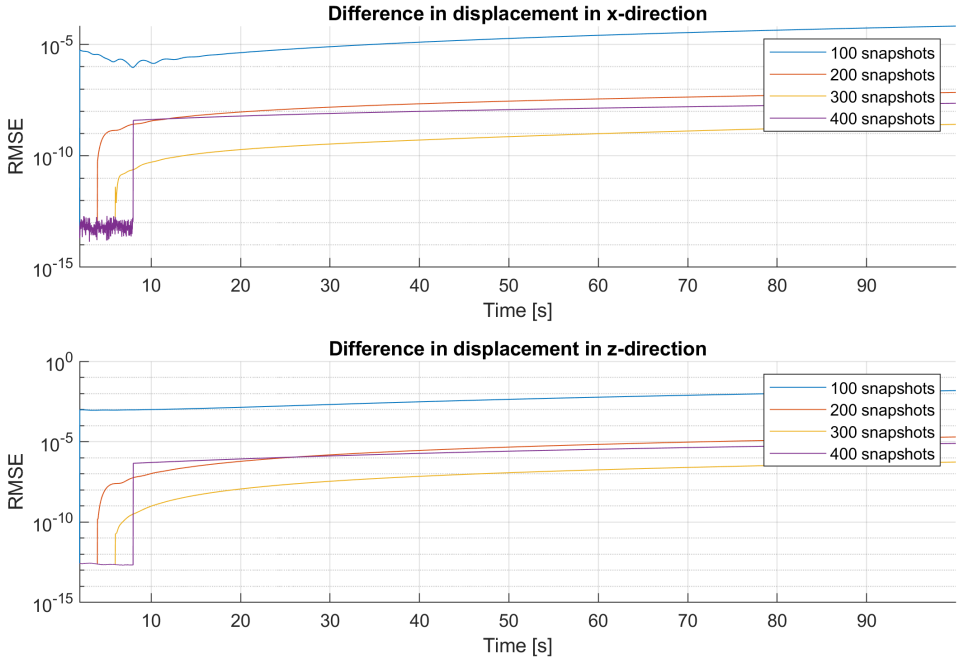


Figure 7.14: RMSE for a DMD reduced order model trained on a single configuration. All the DMD modes are included in this simulation (except for the unstable modes).

Selection of modes

In Figure 7.15, the eigenvalues in the DMD analysis for 400 snapshots are shown. Eigenvalues and eigenvectors for the real matrix A exist in complex conjugate pairs. When λ is a eigenvector with eigenvalue v, $\bar{\lambda}$ is also a eigenvalue with eigenvector $\bar{v}$. This is also visible in Figure 7.15, since it is

symmetric in the real axis. Modes with an eigenvalue with an amplitude larger than 1 will be ignored

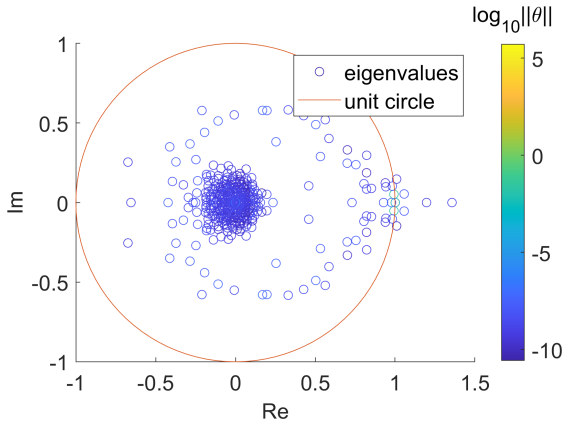


Figure 7.15: DMD eigenvalues in the complex plane of 400 snapshots. The color indicates the norm of the associated scaled mode.

in the reduced solution. When these modes are included, the solution increases exponentially. This is not realistic for this system, because it is not possible that the displacement in z direction or x direction increases exponentially. In Figure 7.14, the unstable modes are ignored as well. The difference between ignoring the increasing modes and including them is shown in Figure 7.16. This figure is in logarithmic scale, so it is clearly visible that when the unstable modes are used, the error increases exponentially.

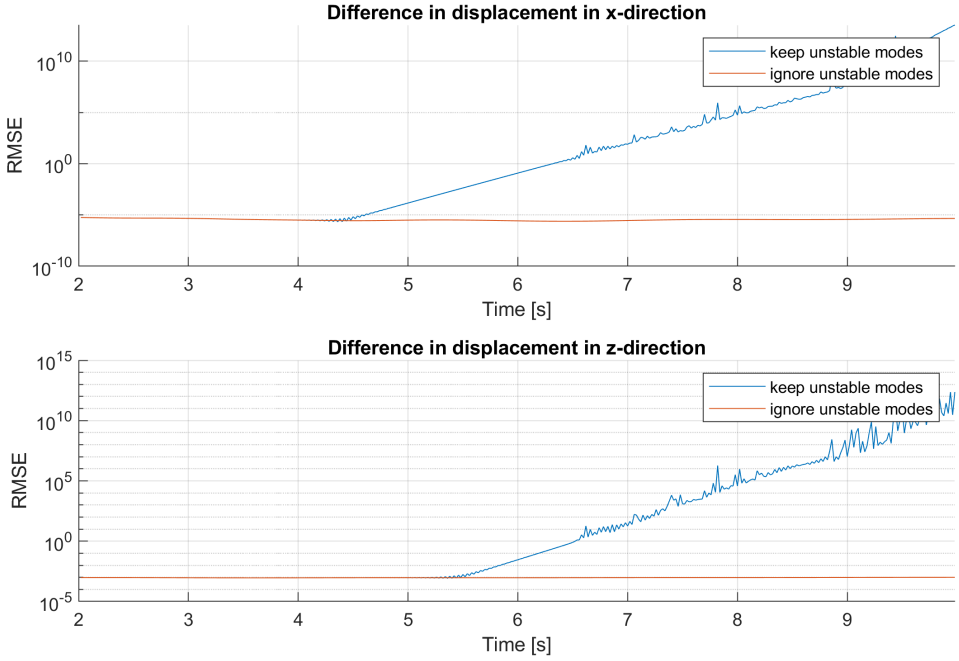


Figure 7.16: RMSE for a DMD reduced order model trained on a single configuration. The unstable modes are either ignored in the solution or included. This result is generated using 100 snapshots to obtain the the DMD ROM.

There are two considerations that determine the complexity of the reduced model.

- The truncation order: this is the number r in Equation 5.21. Instead of computing the eigenvalues of $\tilde{A} = \tilde{U}^\top X' \tilde{V} \tilde{\Sigma}^{-1}$, the eigenvalues of $\tilde{A}$ based on the truncated $\tilde{U}$, $\tilde{V}$ and $\tilde{\Sigma}$ are computed.
- The number of modes included in the solution: after the computation of the DMD eigenvalues and DMD modes, we can choose to not include some of the modes in the solution (Equation 5.30) to reduce the complexity of the ROM further.

By reducing the truncation order, the computational time for the DMD analysis will be reduced. This is because the computation involves finding the eigenvalues of the matrix $\tilde{A} \in \mathbb{R}^{r \times r}$ instead of $A \in \mathbb{R}^{n_{snapshots} \times n_{snapshots}}$. The focus in this research is to develop a reduced order model for long simulations. The time required for this operation is negligible compared to the time needed to generate snapshots using the full order model. Therefore, we have decided not to truncate the singular value decomposition.

Yet, certain DMD modes will be excluded from the solution. Modes with eigenvalues having an amplitude close to 1 do not change significantly over time, so these modes define the steady state and are dominant in the solution for long simulations. Conversely, modes with eigenvalues having an amplitude close to zero only influence the initial phase of the simulation. The influence is also determined by the norm of the scaled mode. Modes with a norm much smaller than the maximum norm of all modes do not affect the solution at any point in time. Given that the primary objective of the reduced order models in this thesis is to perform long simulations, the significance of the modes is determined by the amplitude of the DMD eigenvalues.

In Appendix B, the most relevant modes can be found with their corresponding eigenvalue. Complex modes exist in conjugate pairs, so the mode for one conjugate pair is shown in one figure. Next to that, only the real part is shown. This is the part that will appear in the solution in Equation 5.30, because the imaginary part of one conjugate pair cancels out when the pairs are added in the solution:

$$b_i \phi_i \lambda_i^k + \bar{b}_i \bar{\phi}_i \bar{\lambda}_i^k = b_i \phi_i \lambda_i^k + \overline{b_i \phi_i \lambda_i^k} = 2 \cdot \mathrm{Re}(b_i \phi_i \lambda_i^k). \tag{7.4}$$

The resulting RMSE of the reduced model with 10 modes is shown in Figure 7.17. The plot starts at 8 seconds, because before that time, the full order model is executed to generate snapshots. Therefore, there is no need to simulate the first 8 seconds with the reduced model as well. As mentioned before, the tolerance in this application is 20 µm. Based on the RMSE, we can conclude that DMD gives accurate results, when a proper number of snapshots is chosen.

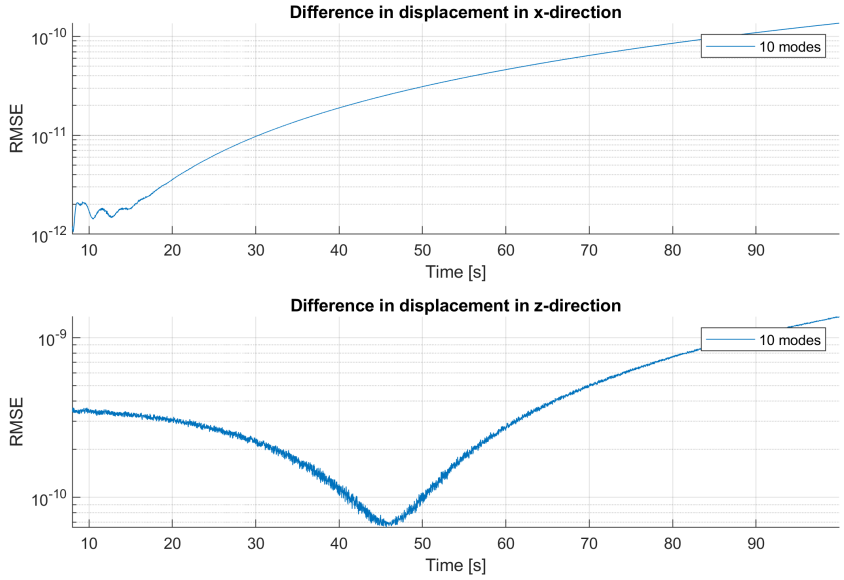


Figure 7.17: RMSE for a DMD reduced order model trained on a single configruation with 400 snapshots.

7.4.2. Computational complexity DMD

After the computation of 400 snapshots, the solution for the remainder of the simulation can be determined using DMD analysis. This analysis involves two computationally intensive operations. Firstly, the SVD of the snapshot matrix must be computed. Secondly, the eigenvalues and eigenvectors of the matrix A need to be determined. However, since the size of this matrix is at most $n_{snapshots}$, this operation is not computationally demanding. For 400 snapshots, these computations together require less than the time needed for one timestep in the full order simulation. Therefore, the DMD analysis will

be neglected. The computational complexity is shown in Table 7.7. Once the DMD analysis is done, the equations needed for the full order model do not have to be assembled. Therefore the function CreateEquationMatrix does not need to be executed anymore. Next to that the function TransportDisplacements only has to be executed for the timesteps where the interpolated displacements are of interest.

Table 7.7: Computational complexity of the DMD ROM for a full system with N unknowns. The number of modes included in the DMD ROM is indicated with r.

Code segment	Dynamic Mode Decomposition
TransportDisplacements	$\mathcal{O}(N)$ (optional)
CreateEquationMatrix	-
SubstituteODE	-
SolveDynamicSystem	$\mathcal{O}(rN)$

7.5. Comparison

In this chapter, POD and DMD were applied on the two-dimensional conveyor belt model with different sets of training data and different number of modes. The summary of the computational complexity is shown in Table 7.8.

Table 7.8: Computational complexity of different methods for a system with N unknowns. The number of unknowns for the algebraic equations is indicated with N_a and the bandwidth of the stiffness matrix is indicated with b. The number of modes included in the ROM is indicated with r.

Code segment	Direct time integration	POD	DMD
TransportDisplacements	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$ (optional)
CreateEquationMatrix	$\mathcal{O}(N)$	$\mathcal{O}(N)$	-
SubstituteODE	-	$\mathcal{O}(N_a)$	-
SolveDynamicSystem	$\mathcal{O}(Nb^2)$	$\mathcal{O}(r^3)$	$\mathcal{O}(Nr)$

In Figure 7.18, the result for solving the system (and substituting the algebraic variables) is shown. With DMD, the resulting displacement for the total simulation is obtained after generation of the training

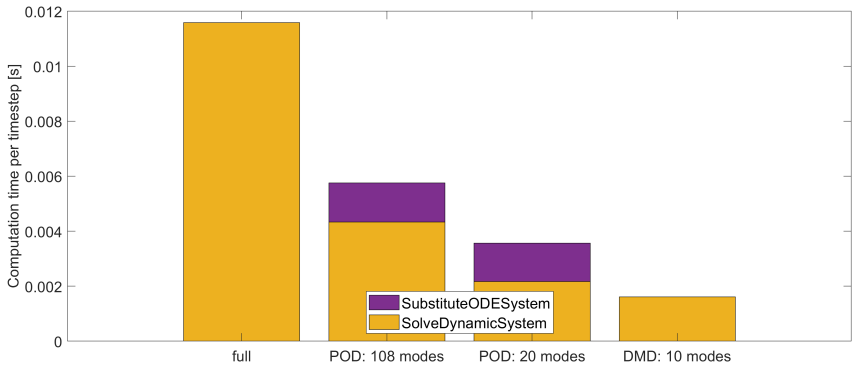


Figure 7.18: The computation time for solving the system in one timestep. The time shown is obtained after generating the snapshots. So generation of the snapshots is not included here.

data and DMD analysis. Therefore, after generation of training data, the result can be obtained with a speedup of 7.2 and for a simulation of 5000 timesteps with 400 modes, this gives a speedup of the total simulation of

$$\frac{5000}{400 + \frac{1}{7.2} \cdot 4600} \approx 4.8. \tag{7.5}$$

POD has the advantage that it can be trained on different data. When the setup of the system of equations and the interpolation step can be avoided or optimized, this gives an advantage for POD. If a broad set of training data is already generated, the full order model does not need to be simulated anymore for extra simulations, but we do need 108 modes for an accurate reduced order model. Therefore, the speedup over the total simulation when training on multiple simulations is 2.0. When POD is trained on the same simulation, we have a speedup of

$$\frac{5000}{400 + \frac{1}{3.3} \cdot 4600} \approx 2.8. \tag{7.6}$$

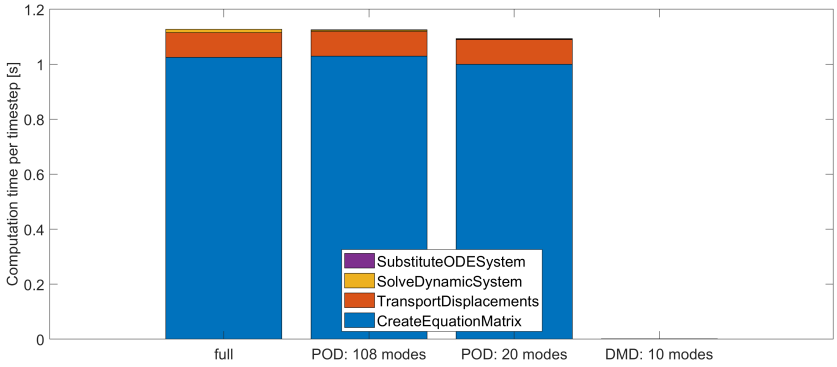


Figure 7.19: The computation time for the different code segments in one timestep. The time shown is obtained after generating the snapshots. So generation of the snapshots is not included here.

The computational time for running the full simulation, after generating the snapshots, is shown in Figure 7.19. POD does not seem to reduce the complete simulation significantly because of the time consuming extra processes. With DMD, however, the time consuming extra processes are included in the ROM and therefore the computation time of running the reduced order model is negligible. In Figure 7.20, the computational time for the functions TransportDisplacements and CreateEquationMatrix with a different number of grid elements is shown. The computational time and the number of grid elements is both plotted in log scale. Because the slope for the computational time of CreateEquationMatrix is around 2, it may suggest a quadratic relation between n and the computational time of CreateEquationMatrix. Therefore, the computational time do not seem to align with the expected computational complexities. Further research is needed to investigate this discrepancy.

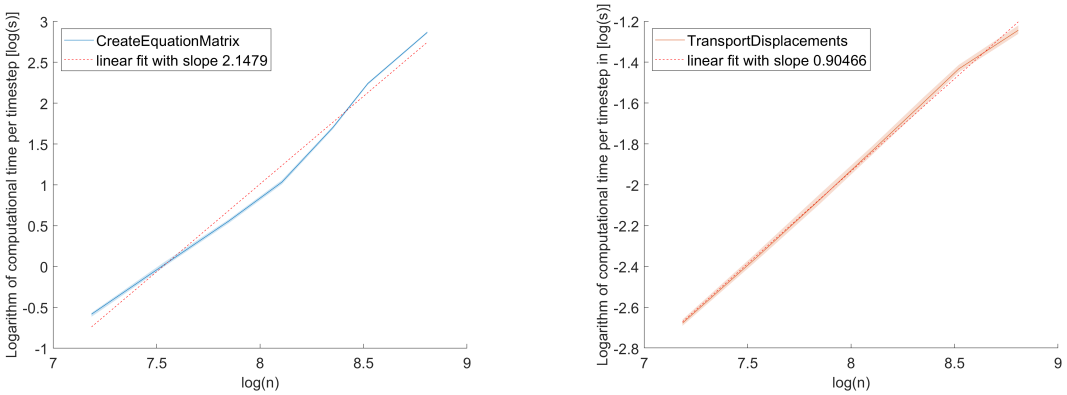


Figure 7.20: Computational time for CreateEquationMatrix and TransportDisplacements for different number of grid elements.

Optimizing the computational time of these functions is beyond the scope of this thesis, but we would recommend to analyze the computational time for these functions as well. Because improving their efficiency would significantly reduce the overall computational time.

Based on the results in this chapter, all methods can generate a sufficient reduced order model with a speedup for solving the system in the time integration. For this, the right training configurations, number of snapshots and modes have to be chosen. The speedup for a system with a grid of 18×104 is of 2 to 5, but for similar systems with more grid points, the same number of modes is needed for an accurate reduced order model. Therefore, the speedup will be higher for systems with the same setup and more grid points.

The same process of training and selecting snapshots and modes is applied on a setup with one tension roller as well, the results are shown and discussed in Appendix C. Because this system behaves more complex than the system examined in this chapter, further research is required to optimize the selection of training data, as well as the appropriate number of snapshots and modes.

8

Conclusion and future research

This study focuses on reducing the simulation of conveyor-belt movement using model order reduction. We started with an existing model for conveyor belt systems. To get an understanding of this model, we derived an analogous model where the belt is modeled as a one-dimensional string instead of a two-dimensional plane in the existing model. Both models involve a DAE to simulate the solution in every timestep. This DAE is different for each time step and is based on the solution of the previous timestep. An equivalent ODE was derived with elimination of variables for model order reduction techniques to be applicable. To correct for movement of the grid points, an interpolation step is applied in each timestep. Every modeling step is the same in both the one- and two-dimensional model, which results in two models with the same structure. Only, in the two-dimensional model, misalignments in three directions can be applied instead of one and the interpolation step is more complex. Nonetheless, because these differences did not interfere the structure of the equations, similar results were expected for applying model order reduction on the two models.

In literature, only one-dimensional belt models are developed and MOR is applied for models with long belt paths. However, the research on two-dimensional belt systems (without transverse displacement) is lacking. In this research, the known theory on the model for a one-dimensional belt system is extended to a two-dimensional belt system by examining an existing model for simulating the movement a two-dimensional belt. The two-dimensional model is studied to answer the following research question:

To what extent can model order reduction techniques accelerate the existing simulations for a two-dimensional conveyor belt system to enable faster detection of issues arising from manufacturing imperfections?

To answer this question, we have addressed the following subquestions.

1. Is it possible to enhance the performance of the existing simulation program for the full order DAEs without yet employing MOR techniques?
2. What is the nature of the nonlinearities in the full order model?
3. What are suitable MOR techniques for accelerating the simulations?
    - In case the systems non-linearities are weak, can we use linear MOR techniques to reduce the the two-dimensional conveyor belt systems?
4. How can we handle the interpolation step when applying MOR?

Is it possible to enhance the performance of the existing simulation program for the full order DAEs without yet employing MOR techniques?

Initially, the solution at each timestep was obtained by forming a single linear system with the dynamical system and the Backward Euler expressions. This approach proved to be time-consuming. We discovered that the process could be accelerated by directly incorporating the Backward Euler expressions

into the dynamical system. This led to a more efficient method for simulating the full order DAE. Furthermore, the algebraic equations within the DAE could be eliminated, resulting in a system of ODEs. However, due to the time-varying nature of the dynamical system, part of the elimination process had to be repeated at each timestep. This step did increase the overall computational time compared to direct time integration with Backward Euler. Consequently, the most efficient method for simulating the full order DAE remained the application of a single time integration method, such as Backward Euler.

What is the nature of the nonlinearities in the full order model?

To apply model order reduction techniques to the system, it is necessary to know the characteristics of the system. In the model, several sources of non-linearities are present. First of all, the radius of the roller that is used in the governing equations can change over time. For rotations in other directions than the moving direction, this radius is not the given radius of the roller, but the length of the node to the rotational axis of the roller. Because of misalignments, this changes over time which introduces non-linearities in the model.

Next to that, the interpolation step is non-linear in the unknowns and results in a changing element mass, which thereby makes changes to the governing equations as well. The result of these three components is that the system is non-linear. However, since the radius and the mass do not rapidly in time, the non-linearities in the governing equations are small.

What are suitable MOR techniques for accelerating the simulations?

After answering the first subquestion, a baseline for the computational time was created. Three model order reduction techniques were explored to further accelerate the dynamic simulation. Because the non-linearities appeared to be small, we focused on linear MOR. Modal decomposition was found to be unsuitable for this type of system, as the modes lack information on the forcing term in the dynamic equations. A novel approach to address this issue was considered but proved inapplicable due to the time-invariant nature of the system of equations that are studied in this thesis.

Secondly, Proper Orthogonal Decomposition was investigated. POD uses simulation data, thereby avoiding the problem of modes not containing information on the right-hand side, encountered in modal decomposition. POD provided accurate results when using the initial iterations of one simulation as snapshot data to reduce the rest of the simulation. Additionally, POD yielded accurate results when simulations from other setups were used as snapshot data. This latter approach is more efficient, as we do not need for full-order simulations to do one simulation. Instead, only a few simulations can be run and saved to obtain a reduced model for various simulations with comparable, but different, configurations. However, one drawback of POD is its applicability to only ordinary differential equations. Because of this, algebraic equations must be eliminated at each timestep. This adds computational time to the simulation of the reduced order model. However, for a low enough number of modes, the total computational time could still improve. Next to that, everything except for solving the dynamical system, could not be reduced with POD. Because the other steps are dominant in the total computational time, the speedup obtained by POD is practically very small.

Finally, the non-intrusive Dynamic Mode Decomposition was applied. DMD is entirely based on simulation data. This has the advantage that every step in the simulation can be included in the reduced order model. DMD trained on initial simulation results provided an accurate solution and was highly time-efficient and the elimination of algebraic equations was unnecessary. However, DMD did not give accurate results when trained on simulations with different configurations. This error could be due to the approximated operation including assumptions about different misalignments present in the snapshot data.

How can we handle the interpolation step when applying MOR?

In the non-intrusive DMD, it was easy to incorporate the interpolation step in the reduced order model. For this, snapshots that account for the interpolation step are used. Modal decomposition and POD, however, could not reduce the interpolation step. In these two methods, the interpolation step was treated separately and only the dynamic DAE was reduced. Therefore, this implementation required switching between the full order state and the reduced order state in each timestep.

The main conclusion of this research is that DMD gives the best reduced model in terms of computational time and accuracy. DMD is able to include all the steps for one simulation in the reduced order

model, in contrast with modal decomposition and POD. By simulating 8 seconds of the belt motion and performing a DMD analysis almost instantaneously, the solution for the remaining 100 seconds of the simulation could be obtained directly with a RMSE in the order of at most 10^{-8}. With POD, an accurate reduced order model could be obtained with a RMSE in the order of 10^{-2} having a significant lower dimension than the full order model. For only solving the system in the time integration, a speedup of 3 to 2 was obtained with POD for a system with a grid of 18×104. And, based on the scalability, this speedup will be higher if a larger grid is used for the same setup. Despite the speedup for solving the system, POD did not result in significant speedup over the total simulation, because of the time-consuming code segments involving the interpolation step and updating the system of equations.

8.1. Future research

Because solving the system in the time integration requires almost negligible time in the full simulation, the focus shifts to reducing the time required for setting up the dynamical system at each timestep (the function CreateEquationMatrix) and the time required for the interpolation step (the function TransportDisplacements). These dominate the overall computational time, computing the time integration is almost negligible. To further speed up the simulations, this setup process should be optimized for computational efficiency. Furthermore, it can be useful to directly formulate the ODE in the setup of the system without generating the full system matrix needed for the naive approach. This approach can inherently incorporate the substitution of algebraic variables within this code segment, so this substitution process does not need to be performed anymore based on the DAE.

If the simulations are still too computational expensive, parallelizing computations in both the setup of the system and the time integration could be considered.

Additionally, the unconventional modeling of the interpolation step gave some challenges in this thesis. The modeling choice of the interpolation step has as consequence that the right-hand side is dependent on the previous solution, which makes the governing equations discrete in time. This results in limited applicability for the standard model order reduction methods, because these are meant for dynamical systems continuous in time. Hence, it is desirable to adopt conventional modeling choices for correcting for movement of the grid points and thereby avoiding the interpolation step. For example with the floating frame of reference approach [28]. A conventional way of modeling will also allow us to analyze the stability of the system better. Next to that, the creation of the equation matrix can be simplified.

This suggestion would allow the application of POD to the model including the interpolation step as well. Alternatively, the interpolation step can be rewritten into a system of equations and could be projected onto a reduced space obtained by POD, to reduce the computational time of the interpolation step. However, with this improvement, the equation matrix still has to be created each timestep.

In modal decomposition, it would be interesting to investigate whether the issue of not accounting for the right-hand side arises in other models as well. A proposed solution to this problem was briefly mentioned in section 6.2 and could be explored further for these type of models.

The last considered MOR method, Dynamic Mode Decomposition, could not be trained on different simulation data to obtain a accurate solution. An option to resolve this, would be to try Dynamic Mode Decomposition with control [44] or parametric Dynamic Mode Decomposition [49].

Finally, in this thesis, the methods were implemented with the number of snapshots determined somewhat heuristically. Further investigation is needed to establish a general procedure for obtaining the number of snapshots required for an accurate reduced model for any type of belt and roller configuration. For example with the error estimators proposed in [46], [47], [48].

Bibliography

[1] "Markets - industrial belting." Calhoun Plastics & Chemicals Inc. https://www.calhounplastics.com/ (Accessed: 07 August 2024). ().

[2] D. Clénet, "Optimising energy efficiency of conveyors," *White paper WP20100601EN, Schneider Electric SA*, 2010.

[3] A. Surtees, "Longitudinal stresses occurring in long conveyor belts during starting and stopping," *Bulk Solids Handling*, vol. 6, no. 4, pp. 93–97, Aug. 1986.

[4] Y. P. D. He and G. Lodewijks, "Green operations of belt conveyors by means of speed control," *Applied Energy*, vol. 188, pp. 330–341, 2017, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2016.12.017.

[5] G. Lodewijks, "Two decades dynamics of belt conveyor systems," *Bulk Solids Handling*, vol. 22, pp. 124–132+173, Mar. 2002.

[6] A. Harrison, "Criteria for minimising transient stress in conveyor belts," *Mechanical Engineering Transactions*, vol. ME8, pp. 129–134, 1983.

[7] G. Lodewijks, "Dynamics of belt systems," Ph.D. dissertation, TU Delft, 1996. [Online]. Available: http://resolver.tudelft.nl/uuid:eed8753e-0c8e-4b9d-95d0-26650f092fa8.

[8] W. Chen and X. Li, "Model predictive control based on reduced order models applied to belt conveyor system," *ISA Transactions*, vol. 65, pp. 350–360, 2016, ISSN: 0019-0578. DOI: 10.1016/j.isatra.2016.09.007.

[9] C. Yang, B. Chen, L. Bu, L. Zhou, and L. Ma, "Low-order dynamical model and distributed coordinated model predictive control for multi-stage belt conveyor systems," *Journal of Process Control*, vol. 124, pp. 83–91, 2023, ISSN: 0959-1524. DOI: 10.1016/j.jprocont.2023.02.010.

[10] G. Suweken and W. Van Horssen, "On the transversal vibrations of a conveyor belt with a low and time-varying velocity. part i: The string-like case," *Journal of Sound and Vibration*, vol. 264, no. 1, pp. 117–133, 2003, ISSN: 0022-460X. DOI: 10.1016/S0022-460X(02)01168-9.

[11] G. Braat, "Theory and experiments on layered, viscoelastic cylinders in rolling contact," Ph.D. dissertation, TU Delft, Jan. 1993. [Online]. Available: http://resolver.tudelft.nl/uuid:5bc3e02e-6d9a-4389-93a5-d1b25c7d36ed.

[12] H. Subbaraman, X. Lin, Z. Pan, *et al.*, "Towards realizing high-throughput, roll-to-roll manufacturing of flexible electronic systems," *Electronics*, vol. 3, pp. 624–635, Nov. 2014. DOI: 10.3390/electronics3040624.

[13] A.-J. Beltman, R. Plak, J. Hazenberg, R. J. Pulles, B. Brals, and G. van Ooik, "Improved ink registration through advanced steel belt steering," *NIP & Digital Fabrication Conference*, 2012.

[14] D. van Eijkeren and R. Idema, "Bandloop - technical documentation," VORtech, Tech. Rep., Sep. 2023.

[15] P. Benner and T. Stykel, "Model order reduction for differential-algebraic equations: A survey," in *Surveys in Differential-Algebraic Equations IV*, A. Ilchmann and T. Reis, Eds. Cham: Springer International Publishing, 2017, pp. 107–160, ISBN: 978-3-319-46618-7. DOI: 10.1007/978-3-319-46618-7_3.

[16] D. Sipp, M. Fosas de Pando, and P. J. Schmid, "Nonlinear model reduction: A comparison between pod-galerkin and pod-deim methods," *Computers & Fluids*, vol. 208, p. 104 628, 2020, ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2020.104628.

[17] A. Alla and J. N. Kutz, "Nonlinear model order reduction via dynamic mode decomposition," *SIAM Journal on Scientific Computing*, vol. 39, no. 5, B778–B796, 2017. DOI: 10.1137/16M1059308.

[18] V. Sonneville, M. Scapolan, M. Shan, and O. Bauchau, “Modal reduction procedures for flexible multibody dynamics,” *Multibody System Dynamics*, vol. 51, pp. 1–42, Apr. 2021. DOI: 10.1007/s11044-020-09770-w.

[19] “Purpose, mission and values.” Canon Production Printing. https://cpp.canon/about/purpose-mission-values/(Accessed: March 07, 2024). ().

[20] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method: Solid Mechanics* (Finite Element Method Series). Butterworth-Heinemann, 2000, vol. 2, ISBN: 9780750650557.

[21] D. Logan, *A First Course in the Finite Element Method*, 4th edition. Thomson, 2007, ISBN: 9780534552985.

[22] U. Ascher and L. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM, 1998.

[23] F. Ebert, “A note on pod model reduction methods for daes,” *Mathematical and Computer Modelling of Dynamical Systems - MATH COMPUT MODEL DYNAM SYST*, vol. 16, Jan. 2007. DOI: 10.1080/13873951003740041.

[24] Z.-Q. Qu, “Static condensation,” in *Model Order Reduction Techniques: with Applications in Finite Element Analysis*. London: Springer London, 2004, pp. 47–70, ISBN: 978-1-4471-3827-3. DOI: 10.1007/978-1-4471-3827-3_4.

[25] N. M. Newmark, “A method of computation for structural dynamics,” *Journal of the Engineering Mechanics Division*, vol. 85, no. 3, pp. 67–94, 1959. DOI: 10.1061/JMCEA3.0000098.

[26] “Runge–kutta methods,” in *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2008, ch. 3, pp. 137–316, ISBN: 9780470753767. DOI: 10.1002/9780470753767.ch3.

[27] R. Courant, K. Friedrichs, and H. Lewy, “Über die partiellen Differenzengleichungen der mathematischen Physik,” *Mathematische Annalen*, vol. 100, pp. 32–74, Jan. 1928. DOI: 10.1007/BF01448839.

[28] C. Nowakowski, J. Fehr, M. Fischer, and P. Eberhard, “Model order reduction in elastic multibody systems using the floating frame of reference formulation,” *IFAC Proceedings Volumes*, vol. 45, no. 2, pp. 40–48, 2012, 7th Vienna International Conference on Mathematical Modelling, ISSN: 1474-6670. DOI: /10.3182/20120215-3-AT-3016.00007.

[29] A. Kröner, W. Marquardt, and E. Gilles, “Computing consistent initial conditions for differential-algebraic equations,” *Computers & Chemical Engineering*, vol. 16, S131–S138, 1992, European Symposium on Computer Aided Process Engineering—1, ISSN: 0098-1354. DOI: /10.1016/S0098-1354(09)80015-X.

[30] P. J. Schmid, “Dynamic mode decomposition of numerical and experimental data,” *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[31] K. Lee and K. T. Carlberg, “Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders,” *Journal of Computational Physics*, vol. 404, p. 108 973, 2020, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.108973.

[32] M. Forgione and D. Piga, “Continuous-time system identification with neural networks: Model structures and fitting criteria,” *European Journal of Control*, vol. 59, pp. 69–81, 2021, ISSN: 0947-3580. DOI: 10.1016/j.ejcon.2021.01.008.

[33] Z. Lai, C. Mylonas, S. Nagarajaiah, and E. Chatzi, “Structural identification with physics-informed neural ordinary differential equations,” *Journal of Sound and Vibration*, vol. 508, p. 116 196, 2021, ISSN: 0022-460X. DOI: 10.1016/j.jsv.2021.116196.

[34] R. Pinnau, “Model reduction via proper orthogonal decomposition,” in *Model Order Reduction: Theory, Research Aspects and Applications*, W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 95–109, ISBN: 978-3-540-78841-6. DOI: 10.1007/978-3-540-78841-6_5.

[35] R. Rodriguez Sanchez, M. Buchschmid, and G. Müller, “Model order reduction in structural dynamics,” Jun. 2016. DOI: 10.7712/100016.2106.9280.

[36] A. Varga, “Balancing free square-root algorithm for computing singular perturbation approximations,” in *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, 1991, 1062–1065 vol.2. DOI: 10.1109/CDC.1991.261486.

[37] P. Benner, S. Gugercin, and K. Willcox, “A survey of projection-based model reduction methods for parametric dynamical systems,” *SIAM Review*, vol. 57, pp. 483–531, Jun. 2015. DOI: 10.1137/130932715.

[38] A. C. Antoulas, R. Ionutiu, N. Martins, *et al.*, “Model order reduction: Methods, concepts and properties,” *Coupled multiscale simulation and optimization in nanoelectronics*, pp. 159–265, 2015.

[39] M. Géradin and D. J. Rixen, *Mechanical vibrations: theory and application to structural dynamics*. John Wiley & Sons, 2014.

[40] J. H. Tu, “Dynamic mode decomposition: Theory and applications,” Ph.D. dissertation, Princeton University, 2013.

[41] J. Kou and W. Zhang, “An improved criterion to select dominant modes from dynamic mode decomposition,” *European Journal of Mechanics - B/Fluids*, vol. 62, pp. 109–129, 2017, ISSN: 0997-7546. DOI: 10.1016/j.euromechflu.2016.11.015.

[42] L. Mentaschi, G. Besio, F. Cassola, and A. Mazzino, “Why nrmse is not completely reliable for forecast/hindcast model test performances,” in *Geophysical Research Abstracts*, vol. 15, 2013.

[43] A. Przekop, X. Guo, and S. A. Rizzi, “Alternative modal basis selection procedures for reduced-order nonlinear random response simulation,” *Journal of Sound and Vibration*, vol. 331, no. 17, pp. 4005–4024, 2012, ISSN: 0022-460X. DOI: 10.1016/j.jsv.2012.03.034.

[44] J. L. Proctor, S. L. Brunton, and J. N. Kutz, “Dynamic mode decomposition with control,” *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016. DOI: 10.1137/15M1013857.

[45] L. Feng, A. C. Antoulas, and P. Benner, “Some a posteriori error bounds for reduced-order modelling of (non-) parametrized linear systems,” *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 51, no. 6, pp. 2127–2158, 2017.

[46] C. Homescu, L. R. Petzold, and R. Serban, “Error estimation for reduced-order models of dynamical systems,” *SIAM Review*, vol. 49, no. 2, pp. 277–299, 2007, ISSN: 00361445.

[47] L. Feng and P. Benner, “Efficient error estimator for model order reduction of linear parametric systems,” in *2019 IEEE MTT-S International Microwave Symposium (IMS)*, 2019, pp. 346–349. DOI: 10.1109/MWSYM.2019.8700642.

[48] D. Xiao, “Error estimation of the parametric non-intrusive reduced order model using machine learning,” *Computer Methods in Applied Mechanics and Engineering*, vol. 355, pp. 513–534, 2019, ISSN: 0045-7825. DOI: 10.1016/j.cma.2019.06.018.

[49] Q. A. Huhn, M. E. Tano, J. C. Ragusa, and Y. Choi, “Parametric dynamic mode decomposition for reduced order modeling,” *Journal of Computational Physics*, vol. 475, p. 111 852, 2023, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2022.111852.

A

POD results: multiple configurations

In this appendix, extra results for POD training on multiple simulations are shown. The test runs are executed with POD trained on the data listed in Table 7.4.

Table A.1: Systematic set of training data used for POD. All training runs have been executed for 50 timesteps.

	Translation roller 1: x^{3D}	Translation roller 1: y^{3D}	Rotation roller 2: x^{3D}	Rotation roller 2: y^{3D}	Rotation roller 3: x^{3D}	Translation roller 3: y^{3D}
Test run 1	2.5 mm	2.5 mm	0.25 mrad	0.25 mrad	0.25 mrad	2.5 mm
Test run 2	-	5 mm	-	-	-	-
Test run 3	-	-	-	0.5 mrad	0.5 mrad	-
Test run 4	12.5 mm	-	-	-	-	-
Test run 5	-	-	-	-	-0.55 mrad	-
Test run 6	2.5 mm	2.5 mm	0.9 mrad	0.25 mrad	0.25 mrad	2.5 mm
Test run 7	2.5 mm	2.5 mm	0.25 mrad	-0.4 mrad	0.25 mrad	2.5 mm
Test run 8	1.25 mm	3.33 mm	0.167 mrad	0.25 mrad	0.1 mrad	5 mm
Test run 9	5 mm	-	0.5 mrad	0.5 mrad	0.5 mrad	-
Test run 10	-	-	1.25 mrad	-	-	-
Test run 11	2.5 mm	2.5 mm	2.5 mrad	0.25 mrad	0.25 mrad	2.5 mm
Test run 12	2.5 mm	-4 mm	0.25 mrad	-0.8 mrad	0.25 mrad	2.5 mm

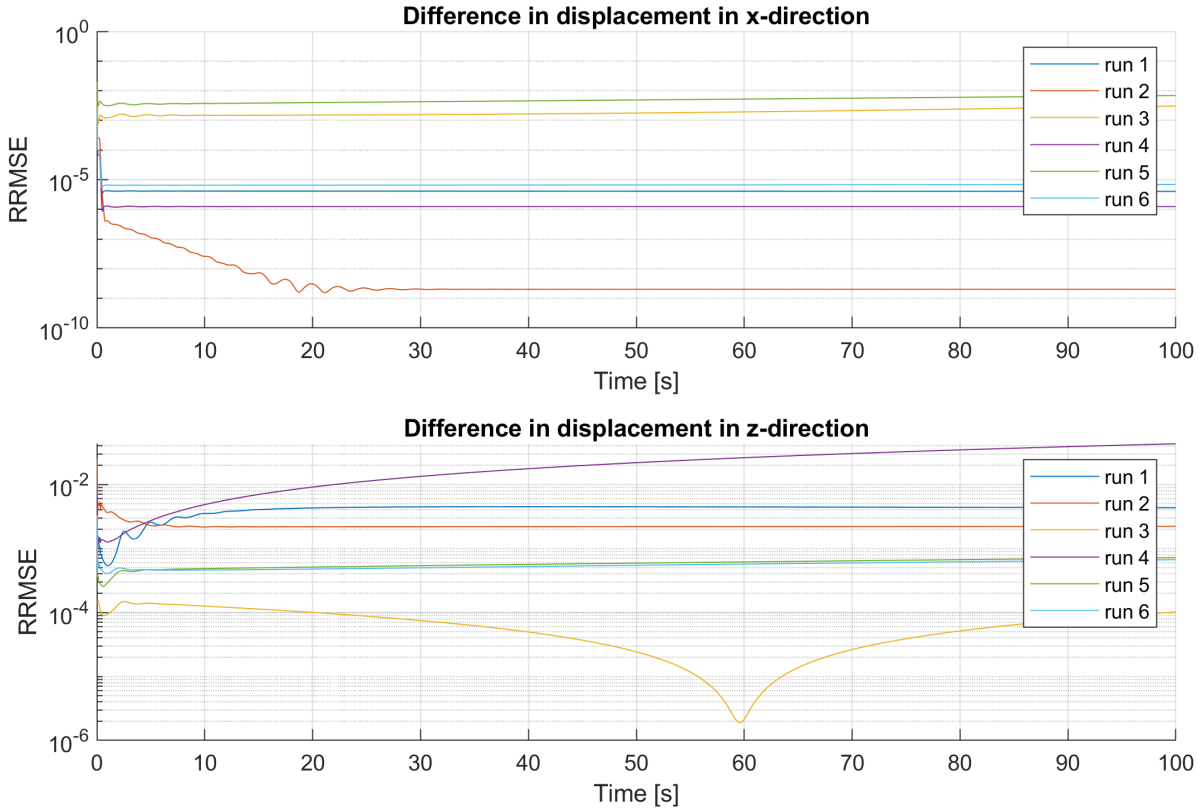


Figure A.1: RRMSE for a POD reduced order model for configurations from Table 7.5 trained on data with multiple configurations from Table 7.4 using 108 modes.

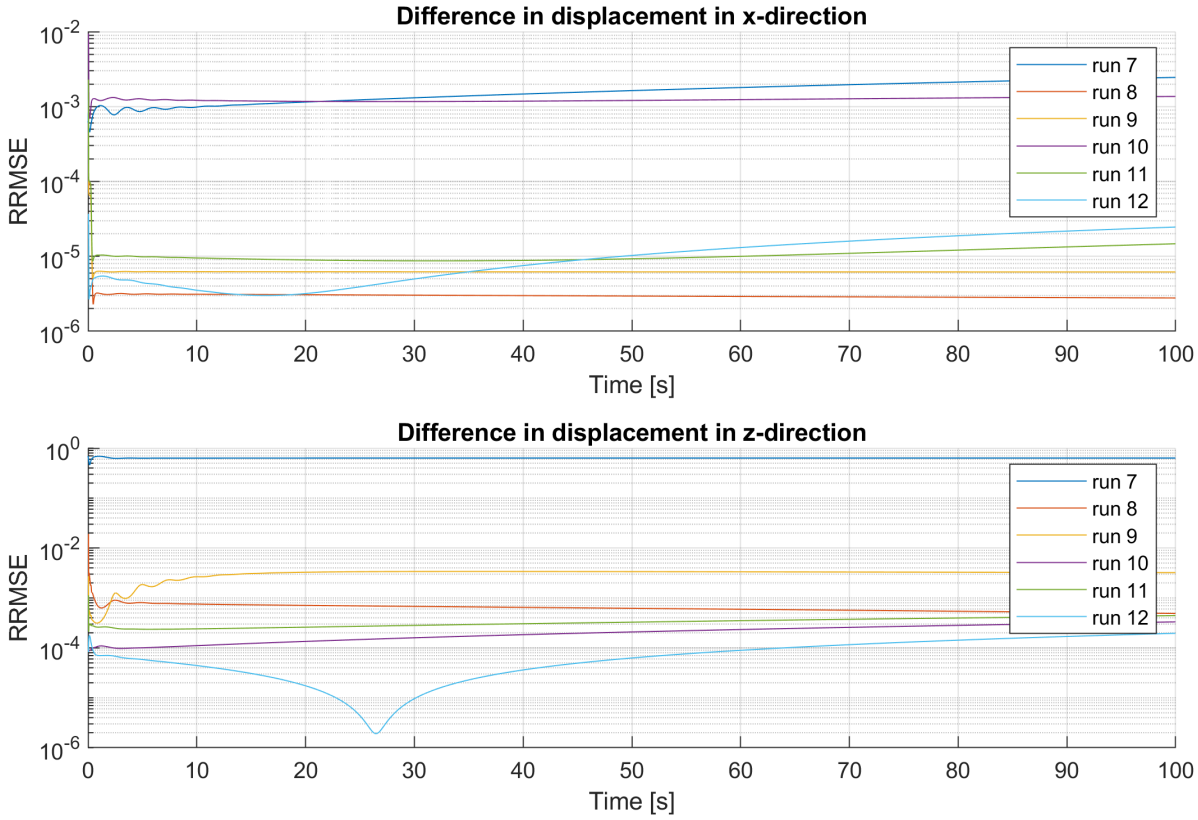


Figure A.2: RRMSE for a POD reduced order model for configurations from Table 7.5 trained on data with multiple configurations from Table 7.4 using 108 modes.

B

DMD scaled modes: single configuration

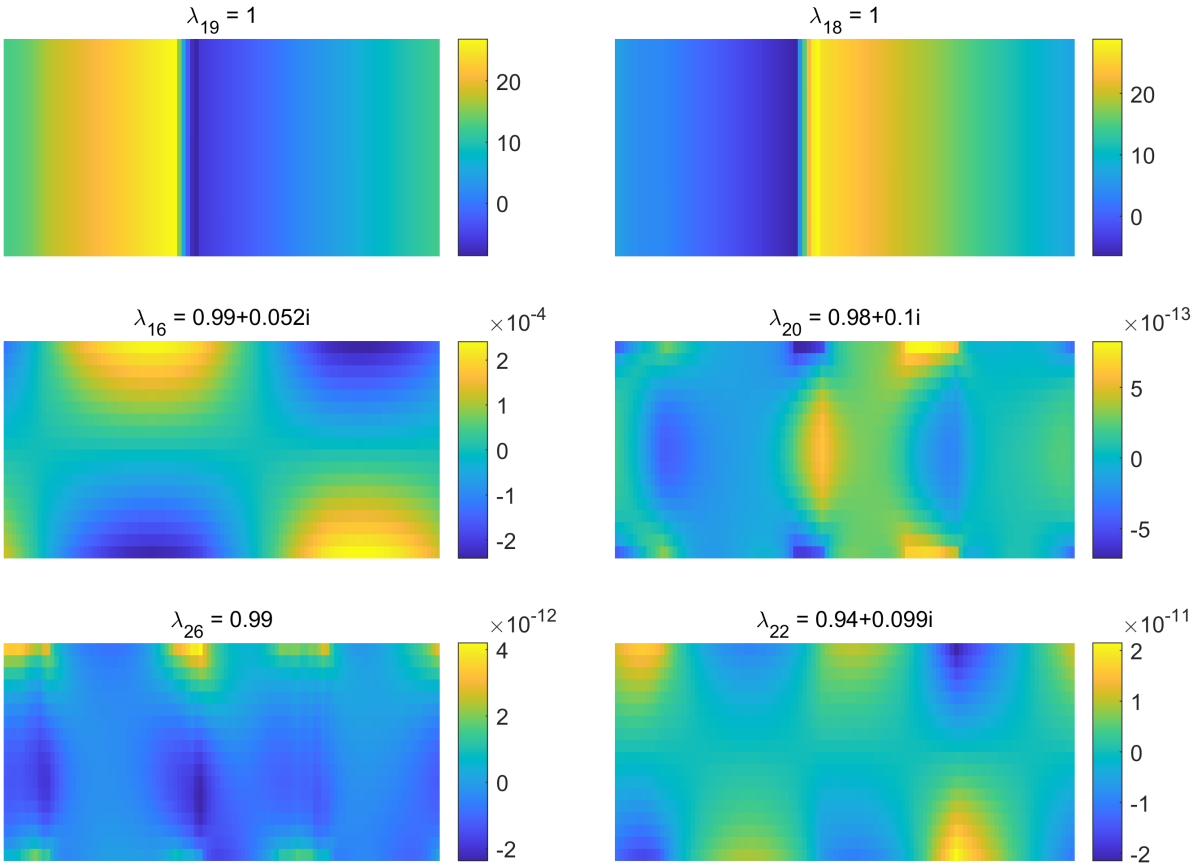


Figure B.1: x displacement of some DMD modes with the corresponding eigenvalue.

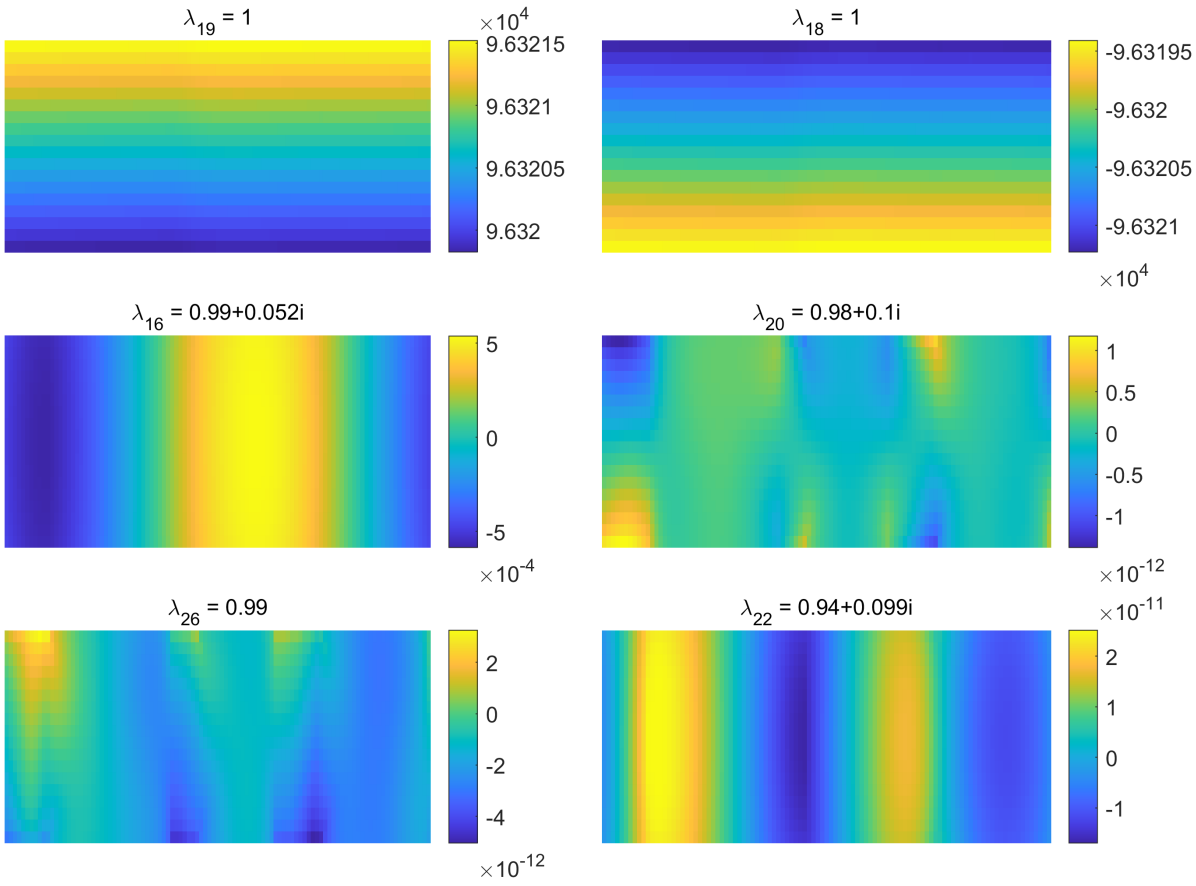
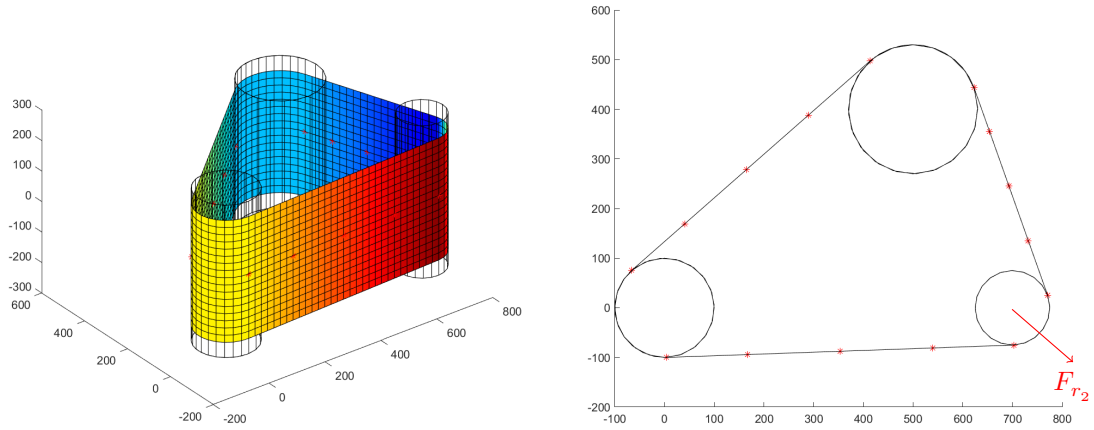


Figure B.2: z displacement of some DMD modes with the corresponding eigenvalue.

C

Results for configurations with a tension roller

The setup is the same as the setup considered in chapter 7, only the second roller is defined as a tension roller. That means there is a force applied on the roller that results in a small tension on the system. The initial state is shown in Figure C.1.



(a) Side view of initial displacement.

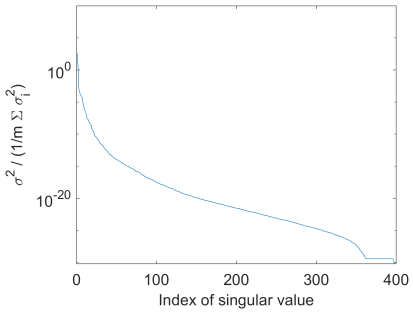
(b) Top view.

Figure C.1: Test setup with a tension roller. Axes are given in millimeters.

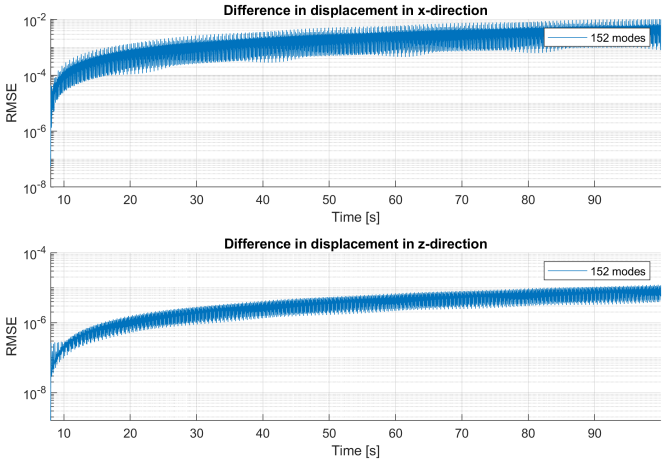
POD and DMD are applied on this setup with configurations listed in Table A.1. The same method is used to select the training data, number of snapshots and modes.

C.1. POD: single configuration

POD trained on a single configuration is applied on run 1 from Table A.1. Based on the singular values depicted in Figure C.2a, 152 modes have to be included. The error is shown in Figure C.2b. The error oscillates with a high frequency. The trajectory of the solution and approximated solution is depicted in Figure C.3 and does also show oscillations. The oscillations in the error can be explained by the oscillations in the trajectory.



(a) Relative singular values of the snapshot matrix based on one configuration using 400 timesteps.



(b) RMSE for a POD reduced order model trained on a single configuration from run 1 in Table A.1 including 152 modes.

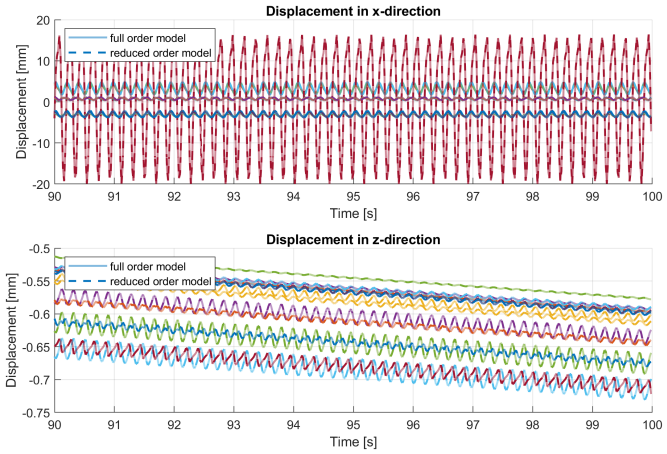
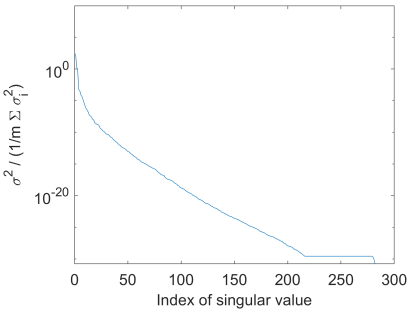


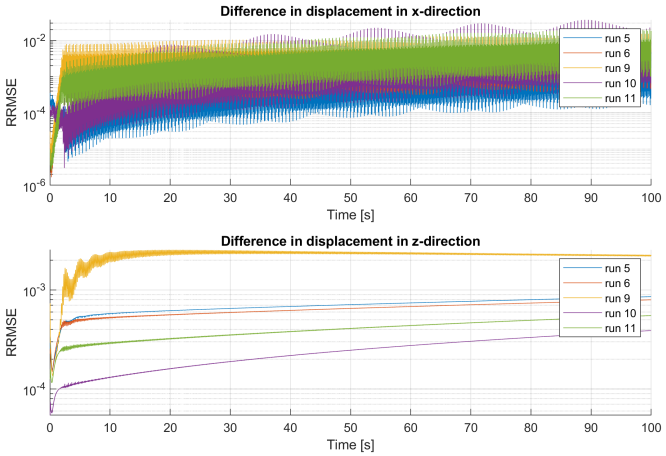
Figure C.3: The trajectory of displacements of tracer points with the FOM and the POD ROM of run 1 in Table A.1.

C.2. POD: multiple configurations

To train POD on multiple configurations, the training data from Table 7.4 and test data from Table A.1 is used. Based on the singular values depicted in Figure C.4a, 113 modes are selected. The error for multiple runs is shown in Figure C.4b. As in POD with a single configuration, this error shows oscillations with a high frequency as well. This is also due to oscillations in the solution trajectory. The error in the x-direction slightly exceeds the precision tolerance. To be certain of an accurate reduced order model, a higher number of modes, or a higher number of snapshots, should be included.



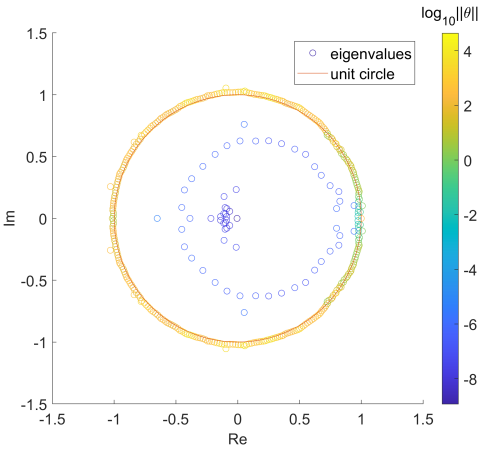
(a) Relative singular values of the snapshot matrix based on the configurations from Table 7.4.



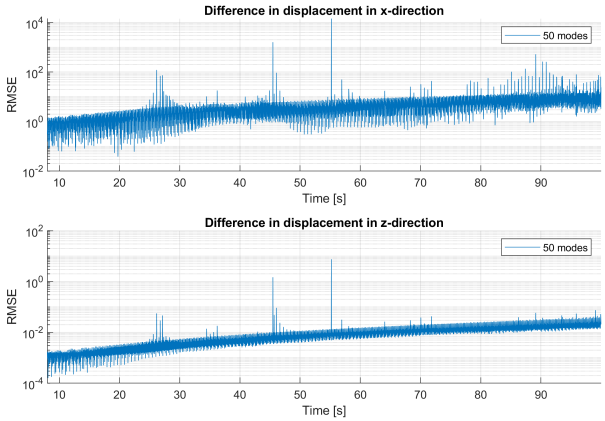
(b) RMSE for a POD reduced order model with configurations from Table 7.5 trained on multiple configurations from Table 7.4.

C.3. DMD: single configuration

DMD trained on a single configuration is applied on run 1 from Table A.1. The eigenvalues are shown in Figure C.5a with the norm of the corresponding scaled mode. A lot of modes have an eigenvalue outside of the unit circle and a high scaled mode norm. When ignoring the modes with $|\lambda| > 1$, and including the 50 modes with the highest eigenvalue, we get a approximation with the error shown in Figure C.5b. The oscillations that have appeared in POD also appear in this method. Next to that, the RMSE is above tolerance in the x-direction and increasing. This gives the impression that, with this number of snapshots and modes, DMD is not able to capture the dynamics of the full order model. Further research can be done on the selection of snapshots and modes in DMD for this setup.



(a) DMD eigenvalues in the complex plane of 400 snapshots. The color indicates the norm of the associated scaled mode.



(b) RMSE for a DMD reduced order model trained on a single configuration using 400 snapshots.