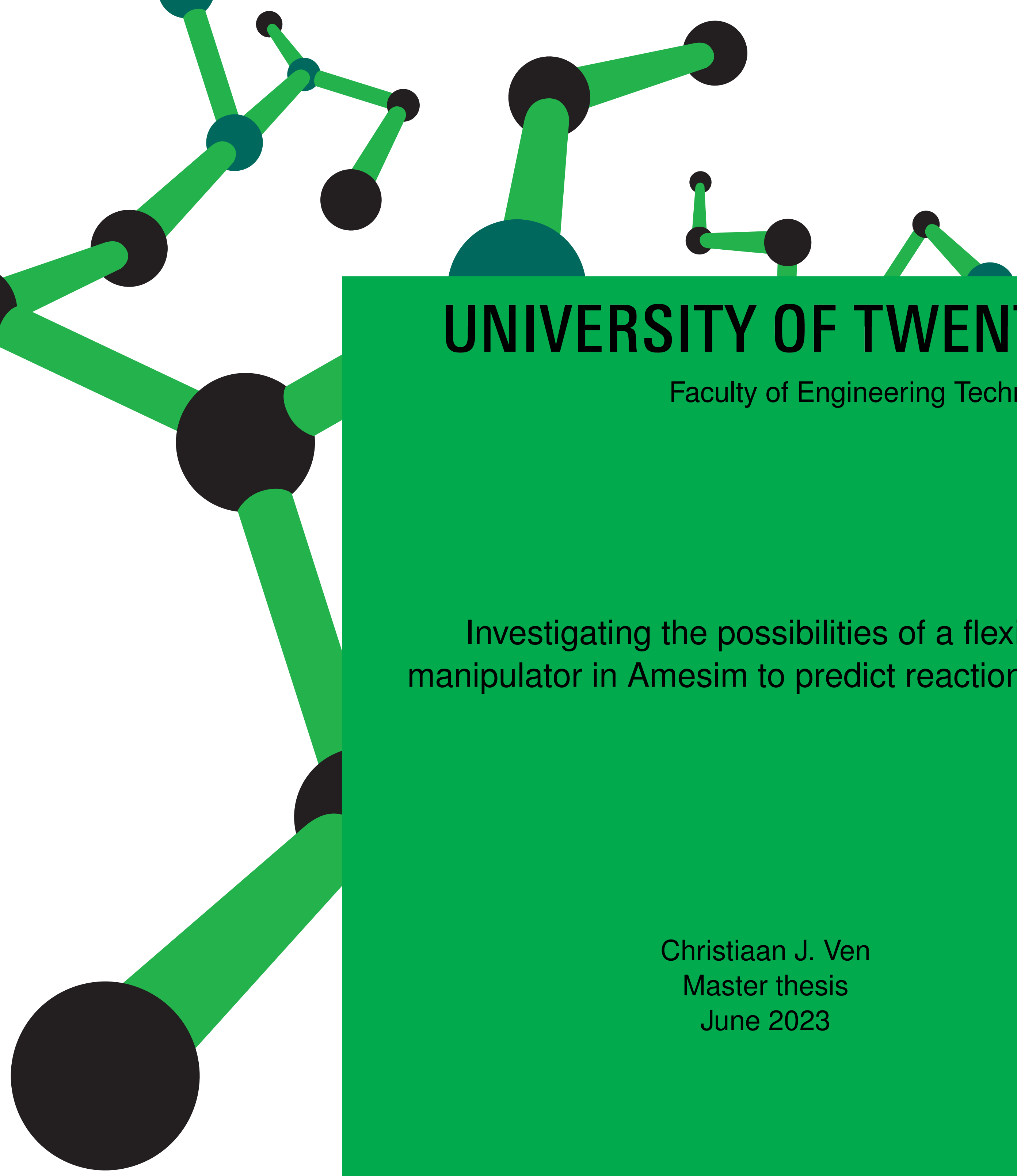


UNIVERSITY OF TWENTE.

Faculty of Engineering Technology

Investigating the possibilities of a flexible manipulator in Amesim to predict reaction forces

Christiaan J. Ven

Master thesis

June 2023

Supervisors:
Dr. ir. J.P. Schilder
Dr. ir. R. Kampinga

Applied Mechanics and Data Analysis Group
Faculty of Engineering Technology
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Contents

1 Summary

Safety is the most important when considering the use of a robotic manipulator. When a robotic manipulator is intended to perform its tasks in a space also inhabited by humans, great care must be taken to prevent any injury. Preventing collisions between the manipulator and its enviroment is the best way to achieve this goal. Since prevention is not always possible, collission detection is also a valuable tool. Detection can be done using dedicated sensors of the manipulator but it can also be done by prediction driving forces and comparing the prediction with measured forces. A difference in these forces might indicate a collision and the control system can then take apropriate actions. For this method to work the reaction forces must be predicted in realtime. When a manipulator can be assumed to behave as a rigid body this prediction is realifly straightforward. For manipulators where, due to lower stiffness, the internal flexibilty contributes significantly to the reaction forces a more complex model is required. The flexibilty of the manipulator is typically determined using a Finite Element Model. These models are large and slow to solve. By reducing the number of degrees of freedom in the model an attempt can be made to implement the flexibilty in realtime. In this research a benchmark problem, a 3D slider crank mechanism, was modeled in Abaqus. A number of reduced models where derived using the substructure functionality of Abaqus. Dynamic simulations where performed of the full order model and the reduced order models. The reduced models where simulated using Abaqus and Amesim. Amesim is a system modeling tool that can also support realtime applications. Comparing the results of all these simulations it can be concluded that Abaqus has issues running a dynamic simulation with a small number of nodes (3 or less). This can be attributed to the choice of computational frame by Abaqus. Amesim requires less degrees of freedom for an accurate simulation. The process of modeling a flexible body simulation in Amesim has still some chalenges but the delevopment is still ongoing.

2 Introduction

Robotic manipulators are becoming increasingly common outside of product manufacturing and the desire to allow interaction between manipulators and humans increases[2]. When robotic manipulators and humans are able to interact great care must be taken to ensure the often powerful actuators do not injure the people interacting with the robot. This can be done by adding sensors that can detect obstacles, i.e. humans, and adjust movement to prevent a collision[3]. Alternatively detection can be done by analyzing the actuation forces, by comparing current actuation forces with predicted actuation forces large differences can indicate a collission[4]. In this research a method for creating a model that can accurately predict actuator forces is investigated. When a robotic manipulator is designed is such a way that the connecting segments, i.e. arms, can be assumed rigid this prediction can be done relatively straightforward[5]. In the case that this assumption cannot be made, because of relatively low stiffness and/or actuation speeds near the eigenfrequency of the manipulator, the flexibility of the connecting elements should be taken into account since this will influence the actuation forces. Flexibility, play and other (non linear) beheviaour in the joint of a manipulator can also influence the reaction forces during movement. The system modeling tool Amesim can be used to take all these factors into account, in this research the focus lies on how such a model should be created and how accurate the results are with respect to a large Finite Element Method (FEM) model.

2.1 IMOCO

This research is part of the Intelligent Motion Control under Industry4.E (iMOCO4.E) project. This is a joint effort of various companies, supported by the European Union, to deliver a reference platform consisting of AI and digital twin toolchains[6]. As part of the IMOCO project the desire to create a Digital Twin (DT) of a robotic arm supporting an x-ray system for the use in a surgical theater, as seen in figure 2.1a, was born. The main goal of this DT is to provide a prediction for the required motor currents needed to maintain or move to a position, by comparing these with the actual motor currents anomalies, collisions, can be detected without requiring additional sensors. Siemens, as one of the IMOCO partners had provided Reden with an AMESIM license to investigate the feasibility of using it for the creation of a Digital Twin.

2.2 Digitial Twin

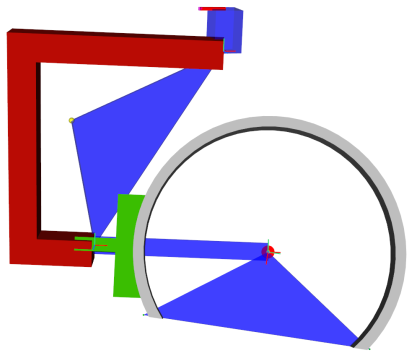
The Digital Twin (DT) concept is very broad and can have various applications, for a broader view on the concept Digital Twin the work of Jones et al. [7] is recommended. In this research a DT is the digital representation of a single, specific, physical device: the Physical Twin (PT). The primary goal of the DT in this research will be to improve the control algorithm and provide additional safety by trying to detect collision by predicting actuation forces. Because of the desired application the DT should be able to perform close to realtime. The DT will initially be created based on a FEM model of the physical device. This DT will never be correct since a model will never be a 100% accurate description of the physical reality. By creating the DT in a parameterized way it should be possible to improve the model during operation: the error of the DT, the difference between expected motor torque and actual, could be used as an input for a parameter fitting algorithm that updates the DT. In an ideal case the only moment that the error is non-zero is when there is an external force applied to the PT, i.e. in the case of a collision. When the error between the DT and the PT is deemed the result of modeling inaccuracy and not caused by a real world disturbance this error could be used to update the DT. By constantly tuning the parameters the model quality should improve over time. This concept is also described by Aivaliotis et al. [8] where a method of creating a DT is proposed. Examples of parameters that are likely to change over time and could be a candidate for updating are: joint friction, joint play and joint stiffness in the constrained directions.

2.3 Amesim

Amesim is system modeling tool that can simulate complex multi-physical systems. One of the toolboxes available can be used to model 3D mechanical linkages and joints. Modeling a system in Amesim is done using components from the various libraries. The 3D mechanical library contains a component that can be used to model a flexible element. One of the advantages of this component is the option to disable the flexibility, this allows for the creation of the total Amesim model by one team when an other team is still working on the creation of the Reduced Order Models (ROMs). Components from the 1D mechanical library can be combined with components from the 3D mechanical library to model non-linear behavior such as play and friction in joints.

The 3D mechanical part allows for the simulation of flexible bodies and non-linear play and friction behavior in joints can be modeled easily. In Amesim it is relatively easy to create a rigid model and enable flexibility on a per-part basis. This allows a team of engineers to parallelize the creation of the total model. A rigid Amesim model can be created and tested while another team member is working on model order reduction of the parts. When a suitable ROM is created this can then be imported in the already functioning model to *upgrade* it from stiff to flexible.

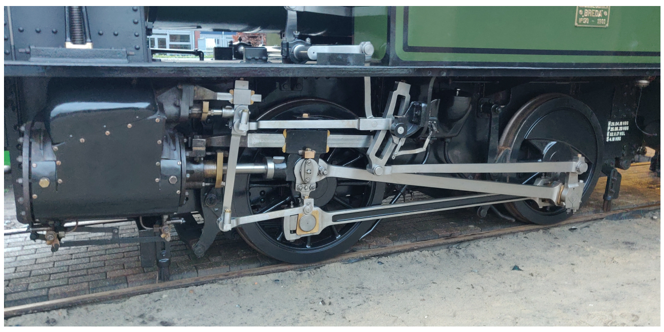
(a) The robotic X-ray machine from Philips[9]

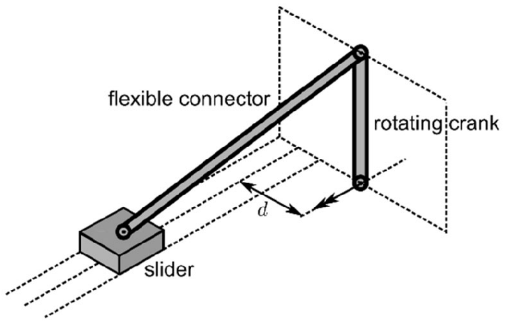

(b) A rigid body model made in Amesim of the x-ray manipulator with 3D-models for visualistion. (blue objects are the rigid body elements and the coloured elements are for visualistion)

The creation of a model of a flexible multibody manipulator in Amesim is split into three distinct phases: the creation of a FEM model of each part of the manipulator, the reduction of the large model to a simpler version and finally the implementation of the reduced model in Amesim. The first two steps will be performed with Abaqus.

2.4 Outline

The first step of this research is to asses how the software that will be used works, In chapter 3 the theory and method of creating a FEM model in Abaqus and how a ROM is obtained from this full order model is discussed. In chapter 4 the process of creating a model in Amesim is described together with the functionality and limitations of a couple of the available components from the mechanical library. A slider crank system is one of the most basic mechanical building blocks, see figure 2.2a for a real world example, therefore it makes sense to use a slider crank as a benchmark problem. The 3D slider crank system from Ellenbroek and Schilder [10], see figure 2.2b, will be modeled and evaluated. The creation


(a) A great example on the application of a slider crank mechanism



(b) The slider crank mechanism used in this research (image from [10])

Figure 2.2: Slider crank mechanisms

and verification of this benchmark simulation is discussed in chapter 5. The results of the various options of a ROM for this slider crank are discussed in chapter 6. Finally the conclusions and recomendations for further research can be found in chapter 7.

3 Modeling in Abaqus

To determine the reaction forces during movement of a robotic manipulator the manipulator must be replicated in a virtual environment. For the creation of such a transient model the software package Abaqus Unified FEA will be used. This finite element analysis program consists of the user interface, to create the model and inspect the results, Abaqus CAE, and the program that applies the FEM to the model and performs the time integration, Abaqus/Standard[1]. Creating the model in Abaqus CAE results in an *input-file*, this file fully describes the model and is provided to the solver. The Abaqus/Standard solver presents the results of the simulation in an output database file which can be viewed using the Abaqus CAE. In this research Abaqus version 2021 HF6 is used

The relation between bodies in motion and forces applied to these bodies is described by Newtons second law of motion[1]:

> Lex II. Mutationem motis proportionalem esse vi motrici impressae, et fieri secundum lineam rectam qua vis illa imprimitur.[2]

Assuming the mass of the body does not change with time, this results in the following equation:

$$\vec{F} = \frac{d\vec{p}}{dt}, \quad \text{with} \quad \vec{p} = m\vec{v} \Rightarrow \tag{3.1a}$$

$$\vec{F} = m\vec{a} \tag{3.1b}$$

3.1 Abaqus theory

In Abaqus Newtons second law of motion is implemented by defining the finite element approximation to equilibrium with equation (3.2).

This differential equation needs to be solved for every timestep to obtain the new configuration of the system. For this a numeric integration algorithm is needed. Generally speaking a choice between an explicit and an implicit time integration scheme must be made. An explicit scheme is faster, per timestep, than a implicit scheme because there is no iteration. However explicit schemes are not conditionally stable and generally require smaller timesteps to remain stable. In this research the implicit time integration is used for all Abaqus models since Abaqus does not support explicit dynamic analyses when substructures are present. In the equations below u^N are the nodal variables.

The equations in this section are taken directly from the Abaqus documentation.

$$M^{NM}\ddot{u}^M + I^N - P^N = 0 \tag{3.2}$$

$$\begin{aligned} M^{NM} &= \int_{V_0} \rho_0 \boldsymbol{N}^N \cdot \boldsymbol{N}^M dV_0 && \text{The consistent mass matrix} \\ I^N &= \int_{V_0} \boldsymbol{\beta}^N : \boldsymbol{\sigma} dV_0 && \text{The internal force vector} \\ P^N &= \int_S \boldsymbol{N}^N \cdot \boldsymbol{t} + \int_V \boldsymbol{N}^N \cdot \boldsymbol{F} dV && \text{The external force vector} \end{aligned} \tag{3.3}$$

[1]There are more solvers available but the Abaqus/Standard is the one used for this research.

[2]Law 2: The alteration of motion is ever proportional to the motive force impressed; and is made in the direction of the right line in which that force is impressed (translated by Andrew Motte, 1846)

3.1.1 Implicit time integration

When using a numerical integration scheme, it is important to determine if numerical damping is required and in what way it should be applied. In Abaqus this is controlled by setting the *application*. The three options are 'Transient fidelity', 'Moderate dissipation' and, 'Quasi-static'. This setting controls various numerical settings such as damping and time incrementation. Quasi-static application is used if the main interest is in the final static response. Transient fidelity has the least numerical damping. With this application small time increments are taken to accurately solve for internal vibrations. The moderate dissipation application is for a wide range of dynamic simulations where the high-frequency vibrations are not of interest. For this research the transient fidelity application is chosen because internal vibrations are relevant for reaction forces.

The Hilber-Hughes-Taylor implicit time integration is used by Abaqus. This results in the transformation of equation (3.2) into

$$M^{NM}\ddot{u}^M|_{t+\Delta t} + (1+\alpha)\big(I^N|_{t+\Delta t} - P^N|_{t+\Delta t}\big) - \alpha\big(I^N|_t - P^N|_t\big) + L^N|_{t+\Delta t} = 0 \tag{3.4}$$

With $L^N|_{t+\Delta t}$ as the sum of all Lagrange multiplier forces associated with Degree of Freedom (DoF) N, and $-\frac{1}{3} \leq \alpha \leq 0$ parameter to control the numerical damping. When α is zero there is the least amount of damping and setting α to a more negative value results in the increased damping of higher frequencies. [11]

$$\begin{aligned} u|_{t+\Delta t} &= u|_t + \Delta t\dot{u}|_t + \Delta t^2\big(\big(\frac{1}{2} - \beta\big)\ddot{u}|_t + \beta\ddot{u}|_{t+\Delta t}\big) \\ \dot{u}|_{t+\Delta t} &= \dot{u}|_t + \Delta t((1-\gamma)\ddot{u}|_t + \gamma\ddot{u}|_{t+\Delta t}) \end{aligned} \tag{3.5}$$

With

$$\beta = \frac{1}{4}(1-\alpha)^2, \qquad \gamma = \frac{1}{2} - \alpha, \qquad -\frac{1}{2} \leq \alpha \leq 0 \tag{3.6}$$

3.2 Full order modeling

In this section the general process of creating a transient simulation in Abaqus is discussed. The reaction forces of a robotic manipulator are highly non-linear with respect to the movement parameters. In a model of a manipulator large (angular) displacements will take place. By enabling *Non-linear geometry* for the Abaqus model these large displacements will be accurately modeled. Although the flexibility of the parts of the manipulator is taken into account, the deformation of the parts is assumed to be small enough to justify linear approximation to the deflections.

Creating parts

The first step when creating a part is to define the geometry, this can be done by using the build in modeling tool or by importing a geometry from a file. Abaqus is unit agnostic, this means that the user should choose a unit system and consistently input the values accordingly. The unit system [mm, ms, g & N] is customary because of the reduced amount of decimal places required for most values. In this research the base SI system of [m, s, kg & N] is used since Amesim uses this system eliminating the need for unit conversion between both programs.

Material properties

When the geometry is completely defined, the part needs to be assigned material properties. Abaqus supports a wide range of (mechanical) material properties, but in this research only linear elastic material behavior is considered. Furthermore, the material properties are considered to be uniform, isotropic and constant. The uniform and isotropic constraint does not influence the generality of this research but mainly simplifies the modeling part. By assuming that the material properties are constant there is a small loss in generality but this assumption will is assumed to hold for most types of robotic manipulators. The material parameters for this research are: the mass density, the Young's modulus and the Poisson ratio.

Meshing

To allow for simulation, the modeled geometry has to be converted to a finite number of elements. These elements must be able to accurately describe the phenomena that occur during the movement of the manipulator. For a large robotic manipulator, such as the one used to support the x-ray machine of Philips bending will probably be the dominant deformation. Pure bending of a beam results in a linear strain field over its cross section. To efficiently represent such a strain field the elements need at least a quadratic interpolation function.

Assembly

In the assembly module of Abaqus multiple bodies can be combined together and the Boundary Condition (BC) and Initial Condition (IC) can be defined. To aid in the definition of these conditions Reference Points (RPs) can be defined. These reference nodes have 6 DoF and can be used to couple nodal DoFs to each other or introduce additional inertia terms.

3.3 Model order reduction

Reducing the model order in Abaqus is done by generating a substructure of the full order model. A substructure is mainly defined by the degrees of freedom required to connect the substructure to other parts of an assembly: the retained DoF. Abaqus uses Guyan reduction[12] to determine a mass and stiffness matrix that only include the retained DoF: the static modes. When the static modes defined by the DoF needed to connect the substructure to the rest of the system are not sufficient to accurately describe the dynamic behavior additional DoF can be added. Additionally, adding generalized DoF associated with the natural modes of the substructure can be added to the system matrices. This method is generally more effective according to section 2.14.1 of the Abaqus theory guide.[13] To obtain the natural modes an eigen frequency analysis can be performed with all retained DoF constrained. This method is called a Craig-Bampton reduction. Both the effect of adding an unconnected interface point and the addition of dynamic eigen modes are investigated.

When a substructure is subject to large rotations Abaqus will determine a rigid body rotation matrix. This is done by picking 3 nodes, starting with the stiffest node (largest diagonal value), the second node will be the one that is furthest away from the first and the third node is the one furthest away from the line between the first two nodes. In the case that there are less than three nodes in the substructure the rotation matrix is determined from the nodal rotations of the stiffest node.

4 Amesim

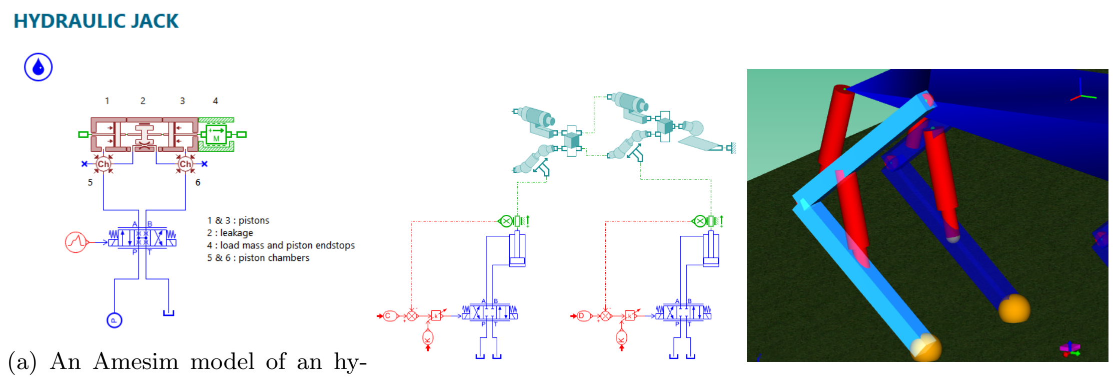
Simcenter Amesim stands for Advanced Modeling Environment for performing Simulations of engineering systems. It is a software package by Siemens designed to create multi physical models on a system level. For this research the 2022.1 version of Amesim is used. Analysing a system with Amesim is divided into 4 phases:

1. Component phase
2. Submodel phase
3. Parameter phase
4. Simulation phase

The component phase is used to add all components of the system to the model. When all components are added and connected to each other the mathmatical models are assigned to the components in the submodel phase. In the parameter phase the properties of the components are set, i.e. mass, locations friction or stiffness. In the parameter phase a study can be created: this allows to simulation to be performed for a range of values of certain parameters without manually editing these values and rerunning the simulation. The simulation phase can start the simulation and allows for plotting and analyzing the results. The assignment of the submodels can be performed automatically if desired.

4.1 Model creation

During the component phase the system is modeled using components available from the various libraries. Every component represents a physical part of the system. A component has a *port* for every interaction that part has with the rest of the system. A *port* can communicate an applied force, postitions (for the mechanical library), voltages, pressures or dimensionless values. Amesim prevents the connection of two components that have incompatible ports.



(a) An Amesim model of an hydraulic jack with leakage and endstops (b) An Amesim model of a robot dog leg with hydraulics (c) The visualization of a robot dog leg

Figure 4.1: Examples of Amesim models from the Amesim help library

4.2 Time integration

According to the Amesim documentation the time integration will be done with either Differential Algebraic System Solver (DASSL)[14] or LSODA. If the model contains any implicit variables, the differential algebraic equation integration DASSL is used, otherwise the ordinary differential equation integration algorithm LSODA is used. There is currently no way of verifying the solver that is used for a specific simulation [1]. This means that any influence on the results of the solver method cannot be investigated. The Amesim documentation mentions that the implementation of both algorithms are substantially different from the original algorithms but does not mention in what way.

4.3 3D mechanical components

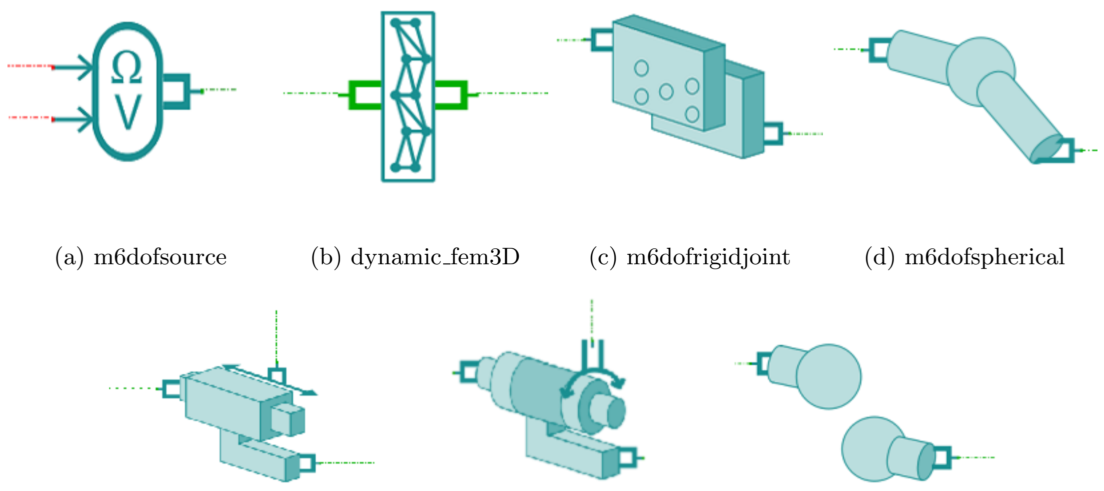
The 3D mechanical library contains components to model mechanical system with 6 DoF. The variable that are communicated over the ports are:

1. absolute position <3> [m]
2. relative position (in local frame) <3> [m]
3. absolute velocity <3> [$\mathrm{m\,s^{-1}}$]
4. absolute acceleration <3> [$\mathrm{m\,s^{-2}}$]
5. Euler angle <3> [degree]
6. rotary velocity <3> [degree/s]
7. absolute angular acceleration <3> [$\mathrm{rad\,s^{-2}}$]
8. frame change matrix (local frame to global frame) <9> [-]
9. force applied <3> [N]
10. torque applied <3> [N m]

For a component with mass (the rigid body component or the flexible body component) the force and torque vectors are defined as input variables and the positional information will be returned as outputs. For the joints the directions are reversed. This means that two bodies cannot be connected directly to each other but require a joint in between.

To manage the 3D mechanical library the *m6dofassembly* component can be added to the sketch. This component can be used to define the gravity vector and to enable or disable the kinematic assembly process. The kinematic assembly is performed before the start of the transient simulation and it changes the orientations and positions of the components to assure the initial configuration satisfies the defined boundary conditions. The *m6dofassembly* also enables the 3D visualization of the simulated system. Bodies are automatically represented as rectangular beams and most joints as spheres. The visualization allows for additional basic geometric shapes to be added to the scene and to change the dimensions, color and visibly of all components. The positions and orientation of manually added objects can be configured to follow variables from the simulation. This feature can be used to create an approximate visualization of flexible bodies since these are not automatically visualized. When a body has complex geometry a step (.stp) file can be imported to represent the 3D shape of the component in the visualization, as seen in figure 2.1b. In this section some of the relevant components from the library will be explained.

[1]Only when the solver encounters issues with stability the used solver is mentioned in the error log.



(a) m6dofsource (b) dynamic_fem3D (c) m6dofrigidjoint (d) m6dofspherical

(e) m6dofpilotedprismatic (f) m6dofpilotedpivot (g) m6dofContactSphereSphere

Figure 4.2: Some of the components present in the 3D Mechanical library of Amesim

4.3.1 Boundary condition component

The *m6dofsource* (figure 4.2a) component can be used to prescribe boundary conditions. The boundary conditions can be positions, velocities or accelerations for the three translational and rotational DoF. The component has a setting to allow for high rotational velocity. When set to high rotation mode the submodel changes the orientation from quaternions to Euler angles. Euler angles have better computational performance but can suffer from gimbal lock. The main axis of rotation needs to be defined before the simulation in this mode, presumably to reduce the change of gimbal lock.

4.3.2 Rigid body component

The *dynamic_3Dbody* is used to model rigid bodies. The coordinates of the ports, interface points, can all be defined in the global frame or in the local frame. In the latter case the position of the origin of the local frame must also be set. The component has inertial parameters (mass, location of Center of Mass (CoM) and, moment of inertia), and parameters to provide the initial velocities. Like the *m6dofsource* component it has a setting for high rotational velocity. The component can be added with two or more ports.

4.3.3 Flexible body component

The *dynamic_fem3D* (figure 4.2b) component can be used to add flexible bodies to a simulation. The dynamic response is split into a rigid body part and a flexible (modal) part. The modal part can be disabled reducing the model to become rigid. This can be very practical from a model debugging perspective since the global system does not change but the simulation cost is decreased greatly. According to the documentation the dynfe component is not suitable for large deformations: ”In case of a flexible body, the deformation angles must be inferior to 20 deg to be in the domain of validity”[2] implying the use of the *Small-angle approximation*. The component can be added with any number of ports.

The computation is defined using two local frames: the i frame attached to the node at port i and the G frame with its origin at the CoM and colinear with the solid frame. The transformation between

[2]From the Amesim help files on the DYNFE001 component

the i frame and the G frame is done with

$$
{}^{G}\mathbb{T}_i = \begin{bmatrix} \left({}^{i}A_s\right)^T & S\left({}^{G}\mathbb{P}_i\right) \\ 0 & \left({}^{i}A_s\right)^T \end{bmatrix} \tag{4.1}
$$

with

- S_a the vector operator such that $S\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$
- ${}^{G}\mathbb{P}_i$ the position vector (3x1) of the port i in the G frame: ${}^{G}\mathbb{P}_i = -{}^{s}P_G +{}^{s} P_i + \left({}^{s}\mathbb{D}_i\right)_{1,2,3}$ with
  - ${}^{s}P_i$ the position vector (3x1) of the port i in the solid frame under rigid assumption, defined by the user.
  - ${}^{s}P_G$ the position vector (3x1) of the gravity center in the solid frame, defined by the user.
  - $\left({}^{s}\mathbb{D}_i\right)_{1,2,3}$ the deformation vector (3x1) computed at port i
- ${}^{i}A_s$ the rotation matrix (3x3) between the solid frame and the port frame
  ${}^{i}A_s = \begin{bmatrix} 1 & \left({}^{s}\mathbb{D}_i\right)_6 & -\left({}^{s}\mathbb{D}_i\right)_5 \\ -\left({}^{s}\mathbb{D}_i\right)_6 & 1 & \left({}^{s}\mathbb{D}_i\right)_4 \\ \left({}^{s}\mathbb{D}_i\right)_5 & -\left({}^{s}\mathbb{D}_i\right)_4 & 1 \end{bmatrix}$

When flexibility is not taken into account the transformation matrix becomes

$$
{}^{G}\mathbb{T}'_i = \begin{bmatrix} I & S\left({}^{G}\mathbb{P}_i\right) \\ 0 & I \end{bmatrix} \tag{4.2}
$$

Rigid body movement

The large displacement, rigid body movement, of the body is calculated separately from the flexibility. The rigid body parameters are the same as the rigid body component with the exception that the flexible body does not have a high rotation mode setting. The equation[3] that is used to determine the rigid body movement is

$$
\sum\left(\left({}^{G}\mathbb{T}'_i\right)^T.{}^{0}\mathbb{F}_i\right) = \mathbb{J}_s.\ddot{\mathbb{X}}_G + \begin{bmatrix} 0 \\ \omega_s \wedge \left(J_s.\omega_s\right) \end{bmatrix} + \mathbb{G} \tag{4.3}
$$

with

- $\mathbb{T}$ the (rigid) coordinate transformation between global and local (body) coordinates
- $\wedge$ the cross vectorial product
- ${}^{0}\mathbb{F}_i$ vector of force (6x1) applied at port i in the absolute frame
- $\mathbb{J}_s$ the inertia matrix (6x6) of the solid at its gravity center $\mathbb{J} = \begin{bmatrix} M_s.I_3 & 0 \\ 0 & J_s \end{bmatrix}$ with
  - M_s mass of the solid
  - J_s rotary inertia matrix (3x3), $J_s = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$ where the matrix elements are defined by the parameters of the component
- ${}^{0}\ddot{\mathbb{X}}$, ${}^{0}\dot{\mathbb{X}}$, ${}^{0}\mathbb{X}$ the acceleration, velocity and position vectors (6x1) of the gravity center in the absolute frame
- $\mathbb{G}$ gravity vector times mass (6x1): $\mathbb{G} = (gx.M_s \quad gy.M_s \quad gz.M_s \quad 0 \quad 0 \quad 0)$ were the gravity vector is defined by the user

[3]Copied verbatim from the Amesim documentation

The acceleration can be determined with

$$^0\ddot{\mathbb{X}}_G = \mathbb{J}_s^{-1}\left(\sum\left(\left(^G\mathbb{T}_i'\right)^T.^0\mathbb{F}_i\right) - \begin{bmatrix}0\\ \omega_s \wedge (J_s.\omega_s)\end{bmatrix} - \mathbb{G}\right) \tag{4.4}$$

The velocity vector is obtained by direct integration:

$$^0\dot{\mathbb{X}}_G = \int_0^t {^0\ddot{\mathbb{X}}_G(u).du} \tag{4.5}$$

The velocity vector is split in a translation part and a rotary part

$$^0\dot{\mathbb{X}}_G = \begin{pmatrix}V_G\\ \omega_s\end{pmatrix} = \begin{pmatrix}V_x\\ Y_y\\ V_z\\ \omega_x\\ \omega_y\\ \omega_z\end{pmatrix} \tag{4.6}$$

The position of the gravity center X_G is computed from a direct integration

$$X_G = \int_0^t V_G(u).du \tag{4.7}$$

The orientation matrix of the solid frame toward the absolute frame $\left(^sA_0\right)$ is computed using quaternions computation:

$$\begin{gathered}\begin{pmatrix}\dot{Q}_1\\ \dot{Q}_2\\ \dot{Q}_3\\ \dot{Q}_4\end{pmatrix} = \begin{bmatrix}0 & -\omega_x & -\omega_y & -\omega_z\\ \omega_x & 0 & \omega_z & -\omega_y\\ \omega_y & -\omega_z & 0 & \omega_x\\ \omega_z & \omega_y & -\omega_x & 0\end{bmatrix}\begin{pmatrix}Q_1\\ Q_2\\ Q_3\\ Q_4\end{pmatrix}\\ ^sA_0 = \begin{bmatrix}Q_1^2+Q_2^2-Q_3^2-Q_4^2 & 2(Q_2Q_3+Q_1Q_4) & 2(Q_2Q_4+Q_1Q_3)\\ 2(Q_2Q_3-Q_1Q_4) & Q_1^2-Q_2^2+Q_3^2-Q_4^2 & 2(Q_3Q_4+Q_1Q_2)\\ 2(Q_2Q_4+Q_1Q_3) & 2(Q_3Q_4-Q_1Q_2) & Q_1^2-Q_2^2-Q_3^2+Q_4^2\end{bmatrix}\end{gathered} \tag{4.8}$$

Modal base

Amesim has an import tool that helps with the setup of the dynfe component. This tool provides two options: *Standard modal base*, where all DoF correspond to physical DoF and, *Super element*, where the DoF are split between physical DoF and virtual nodes that correspond to normal modes. Both options allow importing (proprietary) file types but also direct input of the system matrices.

The import assistant produces two files: a list of eigenvalues and a matrix containing the eigenvectors. These are generated by solving the eigenvalue problem described by the mass and stiffness matrix. The first 6 eigenvalues are practically zero because they represent the rigid body movements. Since the rigid body motion is handled separately, the eigenvalues and corresponding eigenvectors are removed. The entries of the eigenvectors corresponding to these modes are also truncated from the matrix. The matrix is saved such that each nth row in the file contains the values for the nth eigenvalue. The inertial properties and the interface coordinates need to be provided separately by the user. The inertial properties could be extracted from the provide system matrices but this is not done by Amesim. The conversion between the stiffness and mass matrices as performed by the import tool is duplicated using python code as described in appendix D.

When the deformation should be taken into account, the following equations[4] are added to the System of Equations (SoE)

$$\begin{gathered}\begin{pmatrix}\dot{x}\\ \ddot{x}\end{pmatrix} = \begin{bmatrix}0 & I\\ -\Sigma & \lambda\end{bmatrix}\begin{pmatrix}x\\ \dot{x}\end{pmatrix} + \begin{bmatrix}0\\ T\end{bmatrix}.F + \begin{bmatrix}0\\ U\end{bmatrix}.H\\ \begin{pmatrix}d\\ \dot{d}\\ \ddot{d}\end{pmatrix} = \begin{bmatrix}T^T & 0\\ 0 & T^T\\ -T^T.\Sigma & T^T.\lambda\end{bmatrix}\begin{pmatrix}x\\ \dot{x}\end{pmatrix} + \begin{bmatrix}0\\ 0\\ T^T.T\end{bmatrix}.F + \begin{bmatrix}0\\ 0\\ T^T.U\end{bmatrix}.H\end{gathered} \tag{4.9}$$

[4]Copied verbatim from the Amesim documentation

With

- Σ the diagonal matrix containing the eigenvalues (MxM) defined by the user
- T the eigenvector matrix (Mx(6.N)) defined by the user
- λ MxM diagonal matrix containing the eigen damping values $\lambda = 2.diag\big(D_n\sqrt{diag(\Sigma)}\big)$
- U matrix (Mx9) given modal states dependency toward body accelerations, defined by the user.
- H vector (9x1) containing body acceleration and quadratic rotary velocities in the solid frame $H = \begin{pmatrix}\omega_x^2 & \omega_y^2 & \omega_z^2 & \omega_x\omega_y & \omega_x\omega_z & \omega_y\omega_z & {}^s\dot{V}_x & {}^s\dot{V}_y & {}^s\dot{V}_z\end{pmatrix}^T \begin{pmatrix}{}^s\dot{V}_x \\ {}^s\dot{V}_y \\ {}^s\dot{V}_z\end{pmatrix} = {}^sA_0. \begin{pmatrix}\dot{V}_x \\ \dot{V}_y \\ \dot{V}_z - g\end{pmatrix}$
- $x, \dot{x}, \ddot{x}$ modal displacement, velocity and acceleration in the solid frame (Mx1)
- $d, \dot{d}, \ddot{d}$ port deformation amplitude, velocity and acceleration in the solid frame (6Nx1)
- F vertical concatenation of the port forces expressed in the solid frame at the port position (6Nx1) $F = \begin{pmatrix}{}^s\mathbb{F}_1 & {}^s\mathbb{F}_2 & \cdots & {}^s\mathbb{F}_N\end{pmatrix}$ with ${}^s\mathbb{F}_i$ the vector of force (6x1) at port i expressed in the solid frame ${}^s\mathbb{F}_i = {}^s\mathbb{R}_0\,{}^0\mathbb{F}_i, \qquad {}^s\mathbb{R}_0 \begin{bmatrix}{}^sA_0 & 0 \\ 0 & {}^sA_0\end{bmatrix}$

Modal damping

The dynfe component has support for specific modal damping. The user can supply a file containing damping ratios for each eigenmode. In this research this functionality is not further investigated. The default damping ratio (ζ) of 0.02 is applied to all eigenmodes.

Choice of frame by Amesim

According to the Amesim documentation the computation of the deformations are performed in the solid frame. The solid frame is located in the CoM of the body and colinear to the body frame. The body frame is used to define the locations of the interface node, its origin is therefore arbitrary.

Found limitations

The flexible body component has a couple relevant limitations that are (not yet) documented in the help files or indicated by specific error messages. Using the kinematic assembly when a dynfe component is present in the model, even when set to rigid mode only, results in an incorrect result of the kinematic assembly. The assembly will report success but the resulting simulation will not perform as expected even though the initial positions of the interface coordinates will be as expected. This incompatibility was confirmed by the Amesim support team.

When simulating a model, by default, the optimized solver is enabled. According to the Amesim help files using this optimized solver commonly increases computational performance by more than 500%. The documentation does not mention how this optimization operates. However when the optimized solver is not disabled with a dynfe component present, the simulation will fail to run without a specific error message. This incompatibility was confirmed by the Amesim support team.

Manual calculation is needed to determine local body deformations. To determine the difference in orientation of an interface node with respect to the rigid body rotation, a post processing step must be added to subtract the three euler angles at both points from each other. A sensor measuring the orientation at an interface node must be added to obtain this orientation. And, even though according to the help files the local coordinate of an interface node is present on the wire between components, the translational deformation cannot be extracted and should be calculated from the position and orientation of the CoM in combination with the local interface coordinates and the actual global coordinates. This issue was not discussed with the Amesim support.

4.3.4 Joints

The joint components allow bodies to connect to each other or to the fixed world. A joint has positional variables as input and outputs a resulting force. The constraints of all joints can be applied with either an implicit scheme, Baumgarte, or an explicit scheme, spring-damper. In the case of Baumgarte the parameters α and β can be used to ensure the constraints are satisfied. In the case of the spring-damper system, a stiffness and a damping term can be set independently for the translation and rotations.

Actuated (piloted) joints have an additional port to accept a input force (or torque) and output the position and velocity. These ports are compatible with components from the 1D mechanical library to further model the system. For example, a complete hydraulic system including pumps and valves can be included in a model, as showcased in figure 4.1.

Rigid joint

The *m6dofrigidjoint* (figure 4.2c) connects two bodies rigidly and thus has no free DoF. Some issues with this joint where observed that resulted in simulations crashing or stalling at certain points in time. Changing the constraint method from the default implicit to explicit can solve the problem of the simulation crashing. But setting the polar angle to values such as 1, 20, 45 and 89 degrees can also alleviate these issues, for both schemes. This leads to the hypothesis that these problems are the result of gimbal lock.

Pivot

The *m6dofpivot* or *m6dofpilotedpivot*(figure 4.2f) can be used to model a hinge. For both the actuated and not actuated component a spring and damper can be enabled on the free rotation.

Spherical

The *m6dofspherical* joint (figure 4.2d) constraints translation but has all three rotations free. It allows for a stiffness and a damping to be added to the rotations. A related component, the *m6dofconveljoint* (constant velocity), constrains a single rotation. The constant velocity joint does not support stiffness and damping on the free rotations.

Prismatic

The *m6dofprismatic* or *m6dofpilotedprismatic*(figure 4.2e) can be used to model a linear guide. It constrains all DoF except for one translation and supports a stiffness and a damping on the free DoF.

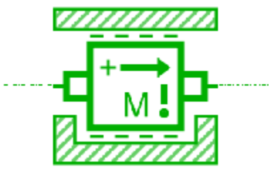
Contact

Modeling contact in Amesim can be done in one of four ways: sphere-to-sphere(figure 4.2g), sphere-to-plane, sphere-to-box, and sphere-to-cilinder. With the exception of the sphere-to-plane the contact can be modeled *internal* or *external*[5]. For the contact friction four models are available: Coulomb friction with hyperbolic tangent, reset integrator, Dahl and, LuGre.

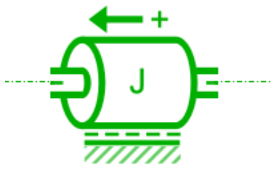
4.4 1D mechanical components

Similar to the 3D mechanical library the 1D mechanical library communicates position and force information over the ports. For masses also the acceleration can be communicated bringing the maximum number of variables per port to four. Combining the components from the 1D library with the 3D library allows for simulation of non-linearities and imperfections of piloted joints. The library is split in components modeling translations and in components modeling rotations. Most translational components have rotational counterparts since the physics are very similar. In this section some relevant components from the library wil be explained.

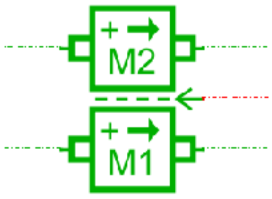
[5]i.e. sphere inside the box or sphere outside the box



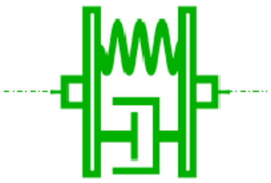
(a) mass_friction_endstops



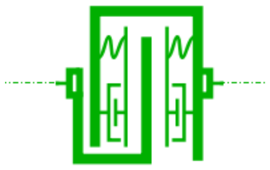
(b) rotaryload2ports



(c) karnop2mass



(d) springdamper01



(e) dblelasticendstop

Figure 4.3: Some of the components present in the 1D Mechanical library of Amesim

4.4.1 Inertia

To model a translating (figure 4.3a) or rotating (figure 4.3b)mass, the inertia components can be used. Both allow for the addition of Coulomb friction. When more specialistic friction models are desired, these need to be modeled using a dedicated component. The translational inertia allows for endstops to be configured directly in the component. There is also the option for a translational inertia with variable mass.

4.4.2 Friction

Modeling friction can be done in the following ways: friction with a stationary surface or with another moving surface, with or without mass. The component for two surfaces with mass (and separate normal force) is depicted in figure 4.3c. To model friction a Karnopp model can be used.

4.4.3 Endstops

The component for linear elastic endstops (figure 4.3e) can be used to model play between two bodies. To ensure continuity in force at the moment of contact the damping coefficient is modified so that it is zero at contact beginning and then approaches its full value asymptotically.

4.4.4 Spring damper

Springs and dampers are one of the most basic 1D mechanical elements. Modeling these with the library can be done separately or combined (figure 4.3d). There are even components with more complex submodels that allow for simulation of rubber material behavior. These rubber components can be configured by either a force-displacement curve or with a Berg model.

4.4.5 Transformers

To interconnect linear and rotational 1D components, various transformers are available, including but not limited to: rack and pinion, cam and follower, slider crack, and a screw-nut component. There are also transformers to couple linear-linear and rotational-rotational motion: worm gears, levers and, linear cam-follower components.

5 Slider crank model

In this chapter the modeling of the slider crank is discussed. The first step is to perform a full order non-linear dynamic analysis as a basis to compare the reduced models to. The absolute quality of this full order model is not the topic of this research since the main interest is the loss of accuracy as a result of the model reduction. Nevertheless, a Floating Frame Formulation (FFF) model of the slider crank will be used to verify the results of the full order model. This is discussed in section 5.2. Since the FFF model does not provide reaction forces the verification will be done based on deflection at the center. The method for model order reduction is described in section 5.3.

In this research the main goal is to determine what the accuracy loss is as a result of reducing the model order. However, verifying the accuracy of the full order model is also relevant. The verification was done using data provided by van Voorthuizen et al.[15]. The code used to generate this data was also used by Ellenbroek and Schilder.

5.1 Abaqus full order model

In this section the full order model of the slider crank is described. In section 5.1.1 the modeling of the flexible connector is described. This part is then placed in an assembly, as described in section 5.1.2, to perform the dynamic simulation. This part is also used in the three model order reduction steps.

5.1.1 The part

The flexible connector is modeled as a cylinder extrude along the (local) z-axis, with a radius of 0.003 and a extrusion length of 0.3. The part is meshed using quadratic hexahedral brick elements with reduced integration (C3D20R elements). The use of solid elements instead of beam elements is to maintain generality. The quadratic interpolation is needed to capture the bending. The total part consists of 720 elements: 12 elements in the cross section and 60 along the length of the bar. Three RPs are defined: along the rotational axis located at both ends and at the center (also the CoM). The RPs are kinematicly coupled to the nodes located at their respective cross sections. The added stiffness as a result of the coupling between the nodes at the cross sections is neglected since the shape of the cross section does not significantly change during bending. Futhermore this added stiffness is present in all Abaqus models of the slider crank. In the paper by Ellenbroek and Schilder the only material properties mentioned are the mass density and the Young's modulus. The python code used to reproduce the results from the paper uses the Young's modulus and a shear modulus. To replicate the same model in Abaqus the Poisson ratio needs to be determined with equation (5.1)[16]. The properties of the part are summarized in table 5.1

$$E = 2G(1+\nu) \tag{5.1}$$

5.1.2 The assembly

The assembly consists of three RPs and the part. The first RP is located at the center of rotation of the rigid crank, with all DoF constrained or prescribed. The second reference point is rigidly coupled, by means of eliminating DoFs, to the first RP. Rotating the first RP around the x-axis results in the circular motion of the second RP in the yz-plane. The third RP is located on the x-axis with only the displacement along this axis unconstrained. This RP has a mass (half that of the slider) associated with

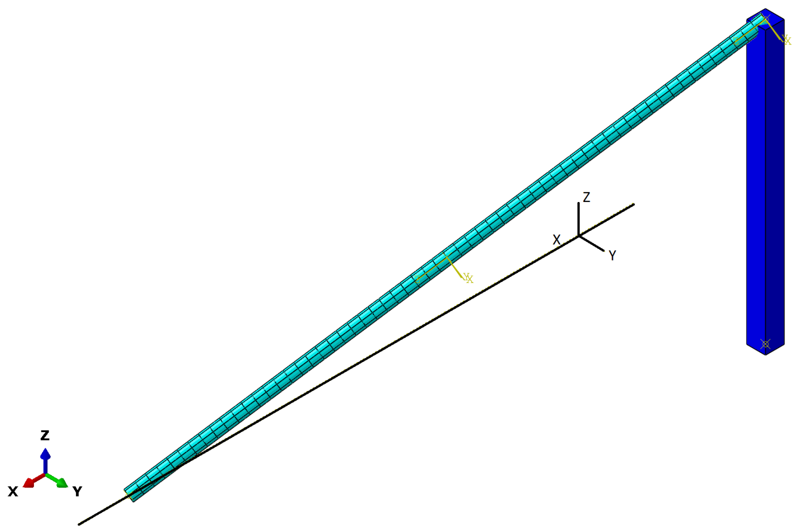


Figure 5.1: Full order model of the slider crank assembly in Abaqus

Property	value	unit
Radius	0.003	m
Length	0.3	m
Young's modulus	200×10^9	Pa
Poisson's ratio	0.299	-
Mass density	7870	$\mathrm{kg\,m^{-3}}$
Resultant properties	value	unit
Mass	0.0668	kg
Moment of inertia around z	3.00×10^{-7}	$\mathrm{kg\,m^2}$
Moment of inertia around x & y	5.01×10^{-4}	$\mathrm{kg\,m^2}$

Table 5.1: Properties of the flexible part of the slider crank mechanism

it. The RPs on the flexible connector are coupled to the RPs of the assembly by constraining only the translational DoFs.

5.1.3 Boundary conditions

To simulate the rotation of the mechanism the rotational displacement of the first RP is prescribed. At the start of the simulation the flexible connector has a initial velocity corresponding with a rigid body rotation of $150\,\mathrm{rad\,s^{-1}}$. To implement this the velocities in x and z direction where determined and linearly interpolated to determine nodal initial velocities. Supplying two different functions for the two different initial velocities is not supported by the Abaqus CAE, so a post-processing step was needed to generate a correct input file. The flexible connector is free to rotate along its axis. This results in one negative eigenvalue of the total system. In the python code used to verify the simulation this rotation is also free.

5.2 Full order verification

The accuracy of the full order model is verified by first reproducing the experiment from [10]: a constant rotational speed of $150\,\mathrm{rad\,s^{-1}}$. Simulating this experiment results in large reaction forces in the first few timesteps. This can be attributed to the implementation of the IC in Abaqus. The simulation code implementing the FFF to perform this validation was provided by K.L. van Voorthuizen and is discussed in section 5.2.1. Because of the high initial reaction forces from the Abaqus model only the center deflection will be compared. To circumvent the issues with this IC a second experiment was introduced with a gradual increase of rotational speed. This gradual increase is implemented with the *smooth step* profile available in Abaqus. In appendix B this smooth step is described and the derivations needed to setup this smooth step in the FFF code.

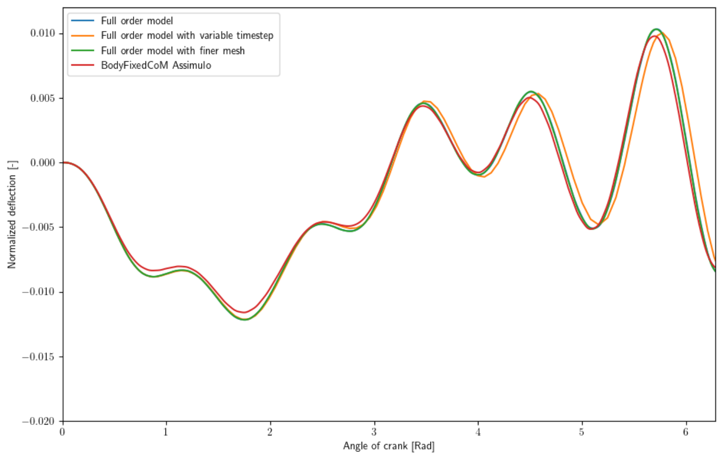
5.2.1 Floating frame formulation

The code used to simulate the slider crank mechanism provided by K.L. van Voorthuizen is written in Python and uses the Assimulo time integration package. In the code the flexible connector is modeled as two bodies. The system matrices for these bodies are created using a Craig-Bamptom reduction. The body starts as a beam with 7 nodes that are reduced to 2 nodes. The reason for the use of two bodies is to allow for the extraction of the location of the center of the flexible connector. The time integration is done using the Runge Kutta34 schema implemented by the Assimulo package[17]. The IC and BC are implemented by prescribing accelerations in the nodes.

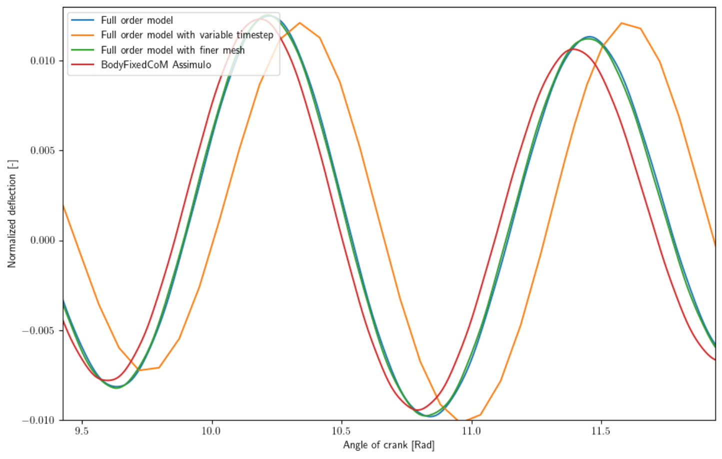
The simulation code should be able to provide reaction forces at the two end points, however these reaction force could not be extracted correctly. All verification is thus done based on the deformation of the flexible connector.

5.2.2 Results

In figure 5.2 the same normalized deflection from [10] is plotted together with three variants of the Abaqus slider crank. The *full order model* is the model with 720 elements with a maximal timestep of 0.06 ms, the *Full order model with finer mesh* is the same model but with 6450 elements, and the *Full order model with variable timestep* has 720 elements but no maximal timestep-size configured. From the deflection plot it can be seen that the timesteps as chosen by the Abaqus solver are too large to capture the deflection. The deflection has a significant phase delay after only a single revolution, as can be seen in figure 5.2b. The version with the same timestep as the python code also shows a phase delay. The cause and solution to this phase delay is not further investigated since optimizing the time integration is not part of this research. Refining the mesh shows no improvement. No further investigation on mesh reduction is done for the slider crank model since this is not deemed relevant for this research.



(a) Deflection during first revolution



(b) Visible phase delay of Abaqus models

Figure 5.2: Normalized deflection of the full order model in three variants and the deflection from [10], with initial rotation BC

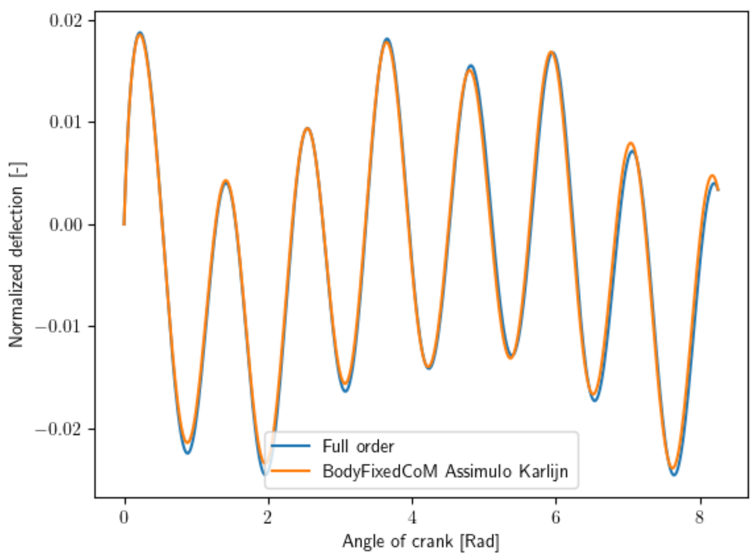


Figure 5.3: Center deflection for full order model and the FFF, with smooth step rotation BC

5.3 Model order reduction

To create the reduced models the full order part is loaded in a separate Abaqus model with a *substructure generation* step. In this model RPs are added in the same way as in the full order model. To allow for the addition of dynamic eigenmodes a *Frequency* step is created to determine these mode shapes. In this frequency step the interface nodes are fixed resulting in fixed-fixed mode shapes.

In total three different reduced model are created: a ROM with only the two interface nodes, a ROM where the centerpoint of the connector is added as a third node, and a ROM that is composed of two substructures, with each 2 nodes, that are joined in the middle. The ROM consisting of two substructures is used to test the hypothesis that having multiple frames, one per substructure, can improve the results.

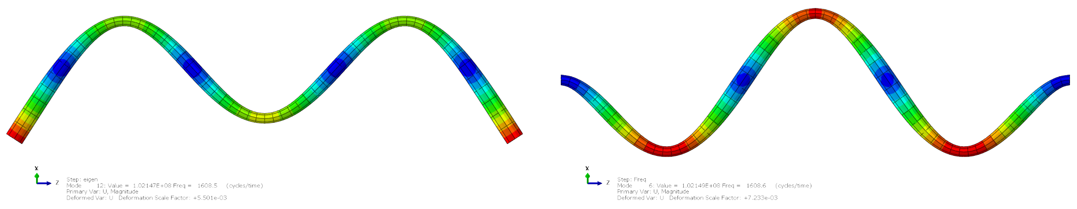
5.3.1 Linear pertubation model

As a first assessment of the ROMs, a linear pertubation model was setup. During such a simulation the choice of frame has no influence on the results. An eigenfrequency analysis was performed on the full order model and on the nine different ROMs. The results of these analyses can be seen in tables A.1 and A.2. The mode shapes of the ROMs were matched to the corresponding modes of the full order model. The free-free mode shapes of the full order model are depicted in figures A.1 to A.11. In general the ROMs have higher frequencies for the highest modes. The ROM will therefore show an increased stiffness in these loadcases.

Looking at the ROM with 3 nodes compared to the ROM the system matrices are very similar. Because the DoFs connecting the two substructures are eliminated the system matrix will be the same as the 3 node ROM. Adding eight eigenmodes to the two substructure ROM will add the same frequency information as adding 16 eigenmodes to the 3 node ROM. Because of the fixed-fixed BC during the eigen analysis every mode occurs at each side of the center node separately. The number of DoF are equal for a 3 node ROM with 16 eigenmodes and a 2 substructure ROM with 8 eigenmodes.

ROM with 2 nodes

Looking at the eigenvalues of the ROM without any eigenmodes added to the system matrix it can be seen that the first two bending modes are significantly higher than their full order counterpart. The first torsional and elongation mode has a higher stiffness than the 6$^{\text{th}}$ bending mode of the full order model. Adding 8 fixed-fixed eigenmodes to the ROM greatly improves the first 8 eigenfrequencies. This



(a) 3rd bending mode free-free (mode 11 & 12) (b) 3rd bending mode fixed-fixed (mode 5 & 6)

Figure 5.4: A comparison of the 3rd bending mode shapes for free-free and fixed-fixed

matches the expectation since the added eight eigenmodes model the same behaviour as the first eight eigenmodes of the full order model. Figure 5.4 shows this for the 3rd bending mode of the connector. The 5th bending mode is now also present in the ROM but with significantly higher stiffness than the full order 5th mode. The eigenfrequency is even higher than the frequency for the 7th bending mode of the full order model. The torsion and elongation modes are not affected by the additional eigenmodes.

Adding 16 fixed-fixed eigenmodes to the ROM results in a correct eigenvalue up until the 7th bending mode. The torsion and elongation modes are not affected.

ROM with 3 nodes

The eigenfrequencies for the ROM with 3 nodes show that just the static modes are enough to describe the 1st bending mode. The other bending modes are significantly stiffer.

Adding 8 fixed-fixed eigenmodes to the ROM has a similar effect as with the 2 node ROM. The next four bending modes are now similar to the full order model. The 6th bending mode is now the first with a higher stiffness.

With 16 fixed-fixed eigenmodes the first eight bending modes are now approximated. The frequencies are closer than the ROM with 2 nodes (and 16 modes) except for the 7th bending mode.

A simulation with 32 fixed-fixed eigenmodes was also performed confirming the hypothesis that the results are the same as the ROM with 2 substructures. The resulting frequencies were omitted from the table.

ROM with 2 substructures

The eigen frequencies of the ROM that consists of two substructures are exactly the same as that of the 3 node ROM when only the static modes are considered. This is to be expected since the resulting model has the same number of DoFs at the same locations with the same BCs.

Adding 8 fixed-fixed eigenmodes to the system matrix results in accurate frequencies up to the 4th bending mode. The higher modes until the 8th bending mode are still close but deviate significantly more.

The variant with 16 fixed-fixed eigenmodes is the first in this set that better describes the longitudinal and torsion stiffness. The first four bending modes are accurate within 1 Hz and the rest of the modes are also quite accurate.

5.4 Modeling in Amesim

The Amesim model of the slider crank can be seen in figure 5.5. A custom smooth-step generator was made to replicate the smooth-step functionality of Abaqus. The rotation is applied using the *m6dofsource* component with a rigidly attached body representing the crank. The flexible body is connected to the crank and the mass with two spherical joints. The joint attached to the mass also constraints the rotation along the center axis of the connector. The mass is modeled as a rigid body and attached via a prismatic joint to the fixed world. Since the kinematic assembly functionality is not compatible with the flexible body component, a dummy model was first created with a rigid connector from which all initial positions and orientations where copied.

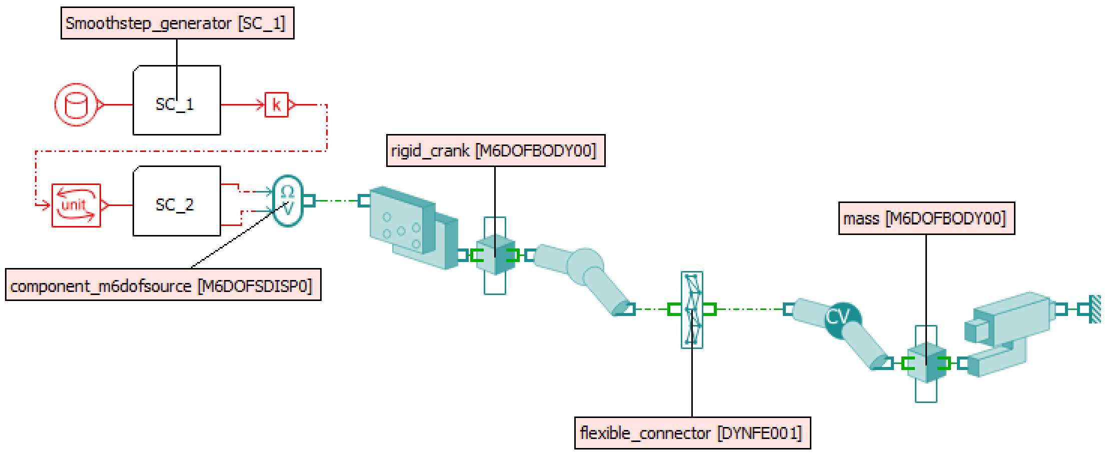
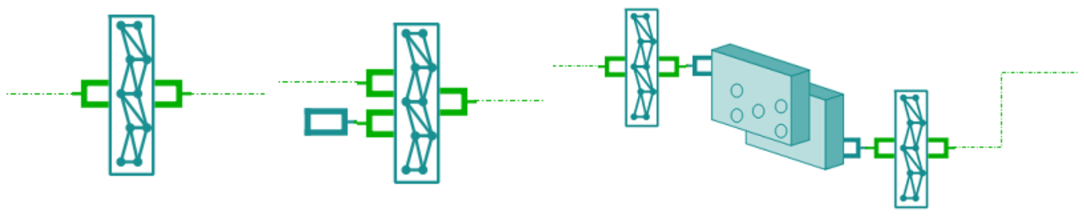

Smoothstep_generator [SC_1]
SC_1
k
unit
SC_2
rigid_crank [M6DOFBODY00]
component_m6dofsource [M6DOFSDISP0]
mass [M6DOFBODY00]
CV
flexible_connector [DYNFE001]

Figure 5.5: Amesim model of the slider crank (ROM with 2 nodes)



(a) ROM with 2 nodes (b) ROM with 3 nodes (c) ROM with substructures

Figure 5.6: The three different ways of including the ROM in an Amesim model

Updating the model to accommodate the three different ROMs can be done by replacing the flexible connector with one of the elements shown in figure 5.6.

For the slider crank model the constraints of the joints had to be modeled with an explicit (spring-damper) scheme. During the simulation of the ROM with 2 substructures, the rigid constraint between the bodies could be solved using the implicit (Baumgarte) scheme for the versions with no and 16 eigenmodes. For the 8 eigen mode model, the rigid joint needs to be solved with the explicit constraint. This initially resulted in poor simuation performance. However, rotating the constrained component by 90° improved the speed significantly. The hypothesis is that this is caused by gimbal lock.

6 Results

In this chapter the results of the experiments with the ROMs are discussed. The goal is to improve the prediction of reaction forces with respect to a rigid body model. In figure 6.1 it can be seen that adding flexibility to the model changes the reaction forces significantly. It also gives an upper limit on the error of the model reduction step. If the error introduced by model reduction is in the same order of magnitude as the difference between rigid and flexible, there is no apparent benefit of simulating the flexibility.

6.1 Comparing reduced order models in Abaqus to full order model

In general, the accuracy of the simulations with the ROMs are low. This can be explained by the way Abaqus determines the computational frame.

6.1.1 Comparing reaction forces

To compare the various ROMs with the full order model (and the FFF code) the different IC from the paper are chosen. The initial rotation leads to large spikes in the reaction forces from the Abaqus simulation. To circumvent this issue a stationary slider is accelerated to $150\,\mathrm{rad\,s^{-1}}$ using the *smooth step* profile available in Abaqus. The Abaqus model imposes the BC by setting the rotational velocity of a RP but the python code requires the acceleration of the point at the end of the rotating crank. The smooth step profile and the derivation of the accelerations are discussed in appendix B.

By combining the difference between flexible and rigid reaction forces with the difference between full order and reduced order, a quick evaluation of the usefulness of a ROM can be performed. In figure 6.2 it can be seen that for all five reaction forces the difference between the ROM and the full order model is similar or larger than the difference with rigid body motion. In figure 6.3 it can be seen that for the most part the error from model reduction is significantly smaller than the difference with rigid body motion. It can also be seen that adding eight eigenmodes does not drastically change the quality of the ROM. Looking at figure 6.5 the use of two substructures leads to high frequency noise in the reaction forces.

Looking at figure 6.4 the errors for the models with eigenmodes are all superimposed. The improvement is mainly gained from adding just eight eigenmodes. Looking at figure 6.6 adding more eigenmodes reduces the high frequency oscillations but it does not reduce the overall error.

Looking at figure 6.7 there seems to be a big difference in model quality between the 3 node and the 2 substructure ROMs. Looking at the reaction forces in figure 6.8 the larger error can partially be explained by a phase difference.

6.1.2 Comparing deformation

Although the prediction of deformations is not the main topic of this research is does provide insight into the accuracy of the ROMs during a transient simulation.

Comparing center deflection

Figure 5.3 shows that the full order model agrees with the FFF on the deflection of the connector at the center for the smoothstep motion profile. Figure 6.9 shows the deflection for the ROM with 3 nodes and

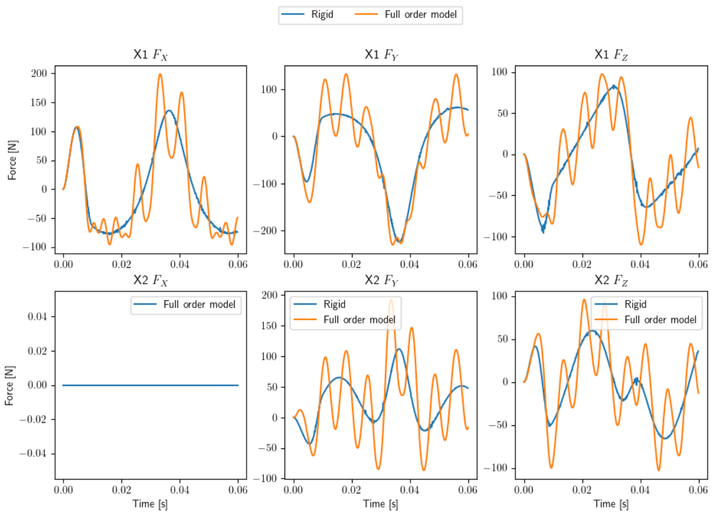


Figure 6.1: Reaction forces at rotation center (X1) and at sliding mass (X2) showing the influence of the flexibility

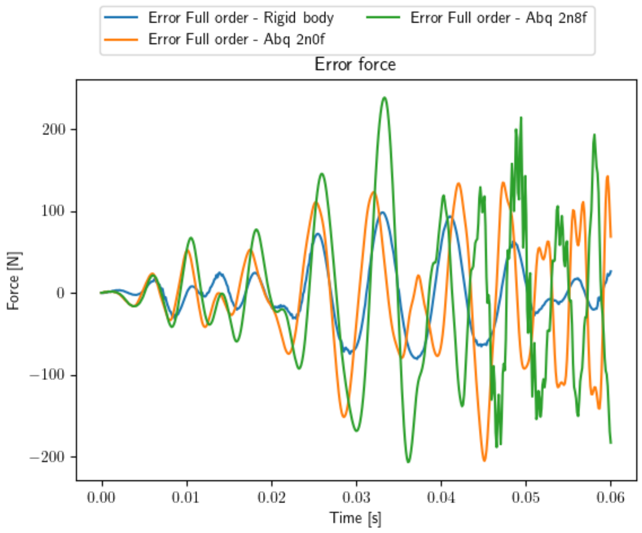


Figure 6.2: Difference in reaction force for full order and: rigid motion, ROM with 2 nodes & no eigenmodes (2n0f) and, ROM with 2 nodes & eight eigenmodes (2n8f)

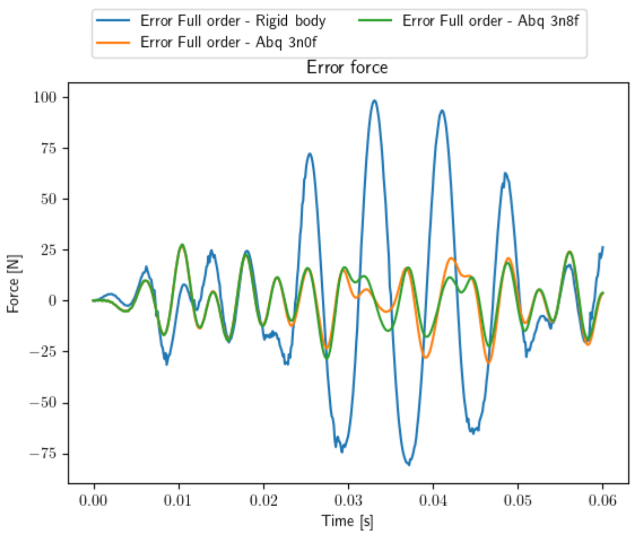


Figure 6.3: Difference in reaction force for full order and: rigid motion, ROM with 3 nodes & no eigenmodes (3n0f) and, ROM with 3 nodes & eight eigenmodes (3n8f)

32 eigenmodes and the ROM with 2 substructures with 16 eigenmodes. In terms of DoF these models are the same. However the deflection of the ROM with 3 nodes is closer to the full order model. It must be noted that for the center deflection the influence of adding eigenmodes is negligible. This can be seen in figure E.1. In the case of the ROM with 2 substructures the addition of eigenmodes seems to decrease the accuracy of the deflection as seen in figure E.2.

6.2 Comparing reduced order models in Amesim to full order model

Performing the simulation in Amesim shows promising results. The accuracy is higher but the influence of the additional modes is lower.

6.2.1 Comparing reaction forces

The ROM with two nodes and no eigenmodes does not contain enough information to accurately determine the reaction forces as can be seen in figure 6.10. Adding the first eight eigenmodes greatly improves the quality of the reaction forces. Figure 6.11 shows the ROMs with 8 and 16 eigenmodes following the full order model quite well. The figure also shows that the accuracy of this ROM is higher in Amesim than in Abaqus. Plotting the reaction forces for all ROMs[1] in figure 6.12 shows a distinct difference between the ROMs with two substructures and the single substructure ROMs. Looking at figure 6.13 there is negligible improvement by adding more nodes or eigenmodes.

6.2.2 Comparing deformation

Looking at figure 6.14 it can be seen that the Amesim model is a bit stiffer than the full order model. And there is no significant effect of adding dynamic eigen modes. Figure 6.15 shows that the deflection, just as the reaction forces, show totally different behavior than the full order model.

[1]Except the ROM with 2 nodes without eigenmodes

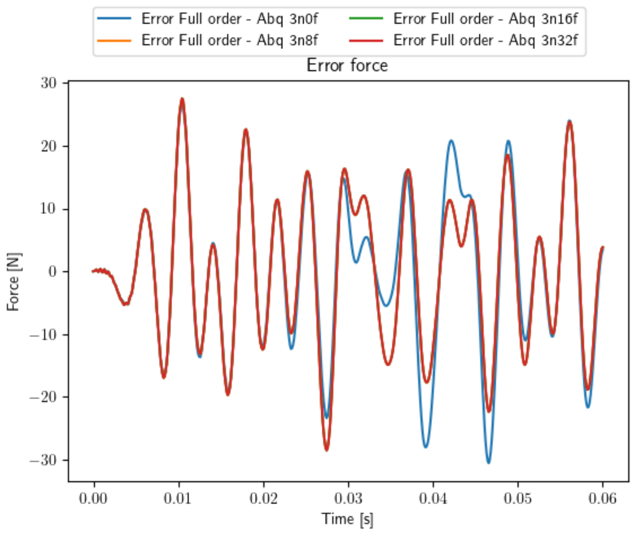


Figure 6.4: Difference in reaction force for full order and: a ROM with 3 nodes without eigenmodes (3n0f) and with 8, 16 and 32 eigenmodes (3n8f, 3n16f and 3n32f respectively)

		Execution time [s]	
		Abaqus	Amesim
2 nodes	0 modes	80.1	8.1
	8 modes	81.8	27.5
	16 modes	83.6	161.7
2 substructures	0 modes	98	7.04
	8 modes	101	101.234[2]
	16 modes	99	31.0313
3 node	0 modes	75.6	21.21
	8 modes	75.39	68.75
	16 modes	75.4	344.516

Table 6.1: Total cpu time in seconds for the simulation of 0.06 s slider crank motion

6.3 Computational performance

The duration of all simulations for the various ROMs in Abaqus and Amesim are combined in table 6.1. When simulating in Abaqus the addition of additional DoF does not significantly increase the execution time. This implicates a bottleneck independent on the number of DoF. The high frequency noise present in the results of the 2 substructure ROM is probably the cause of the higher execution time. The 2 node ROM also shows high frequency noise (see figure 6.2), the 3 node 16 eigenmode ROM had over four times more DoF than the 2 node 0 eigenmodes ROM but is still faster to simulate.

[2]this model is slightly different than the 0 and 16 mode variants (as discussed in section 5.4)

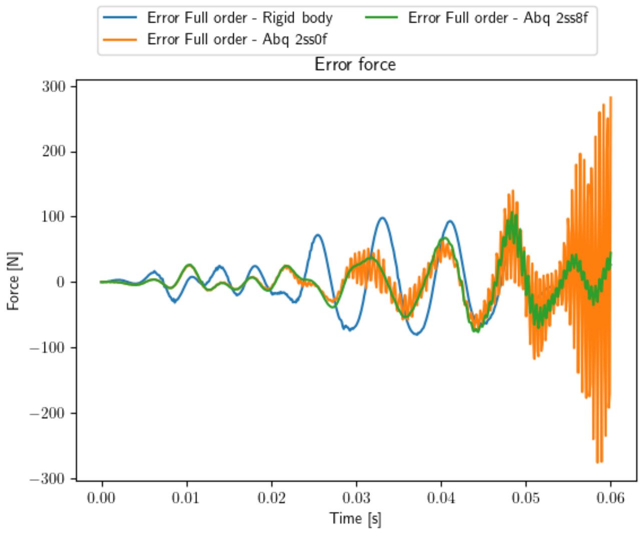


Figure 6.5: Difference in reaction force for full order and: rigid motion, ROM with 2 substructures & no eigenmodes (2ss0f) and, ROM with 2 substructures & eight eigenmodes (2ss8f)

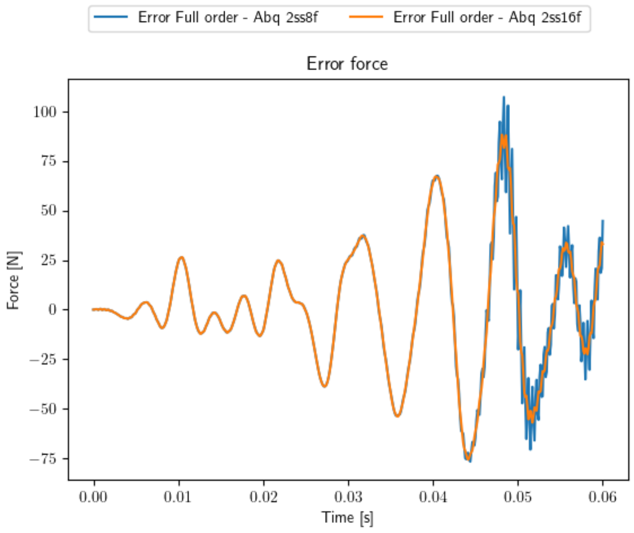


Figure 6.6: Difference in reaction force for full order and: a ROM with 2 substructures without eigenmodes (2ss0f), with 8 eigenmodes (2ss8f) and 16 eigenmodes(2ss16f)

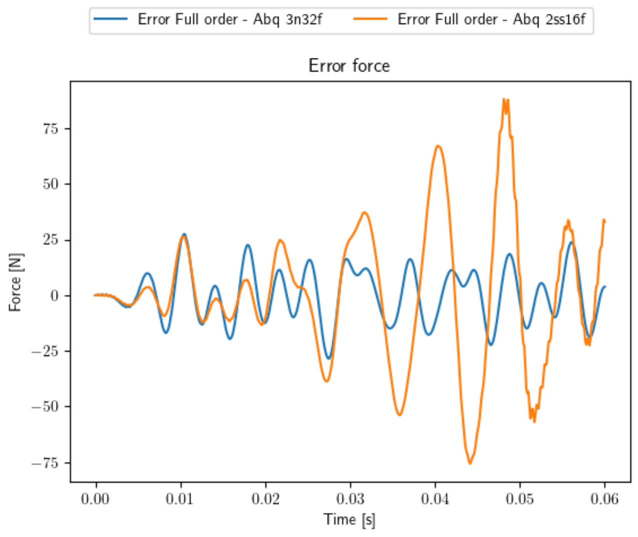


Figure 6.7: Difference in reaction force for full order and: a ROM with 2 substructures with 16 eigenmodes (2ss16f) and a ROM with 3 nodes and 32 eigenmodes(3n32f)

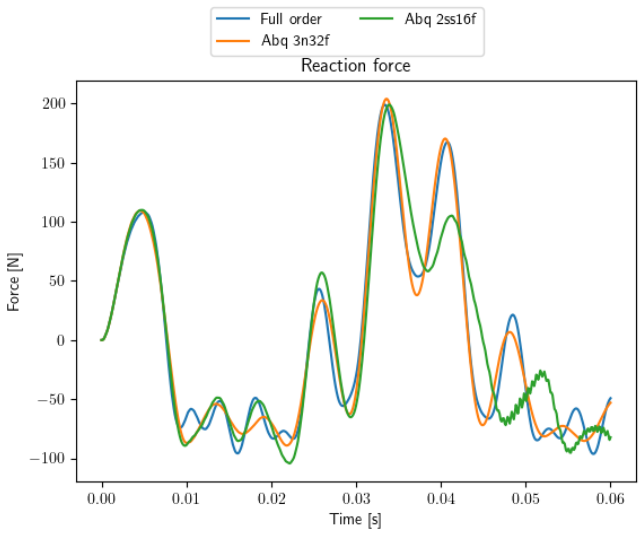


Figure 6.8: Reaction force for full order, a ROM with 2 substructures with 16 eigenmodes (2ss16f) and a ROM with 3 nodes and 32 eigenmodes(3n32f)

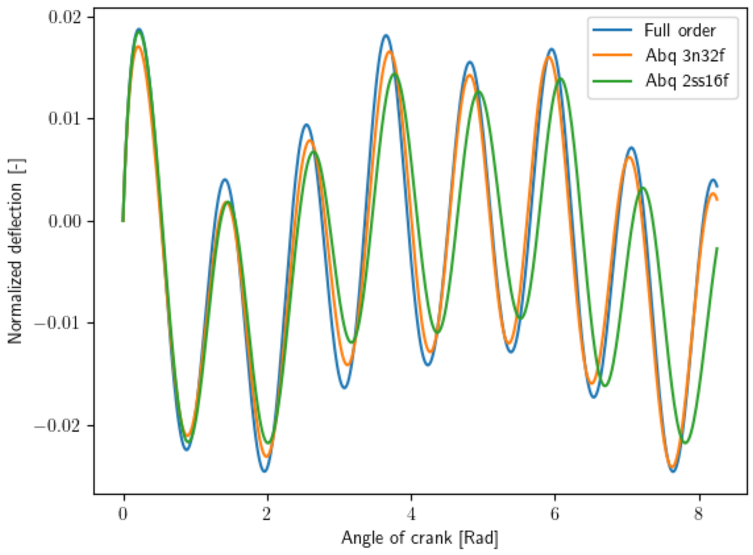


Figure 6.9: Center deflection comparing ROM with 3 nodes and 2 substructures

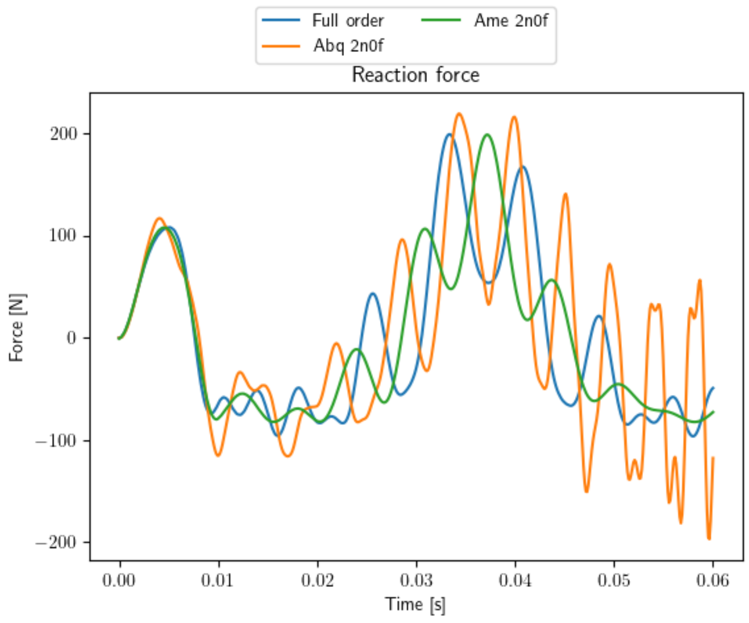


Figure 6.10: Reaction forces at the top of the crank in X direction for the full order model and for the ROM with 2 nodes and no eigenmodes

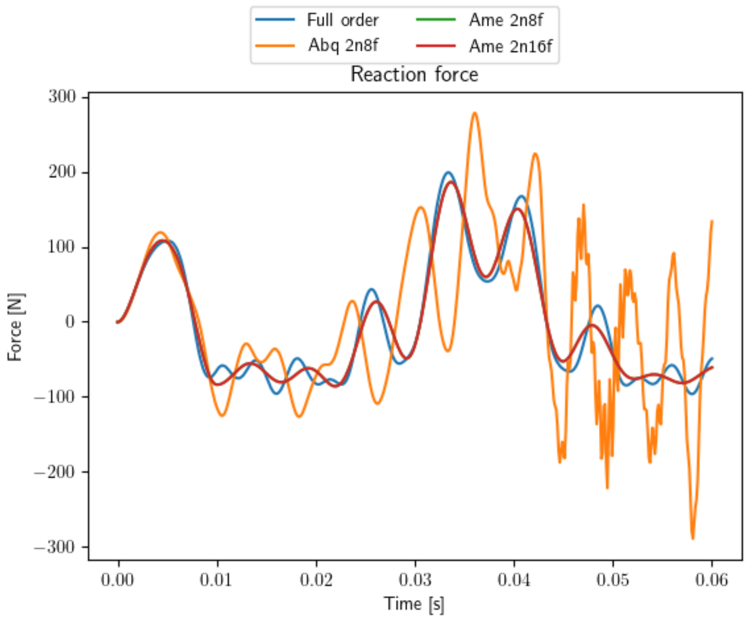


Figure 6.11: Reaction forces at the top of the crank in X direction for the full order model and for the ROM with 2 nodes and no eigenmodes

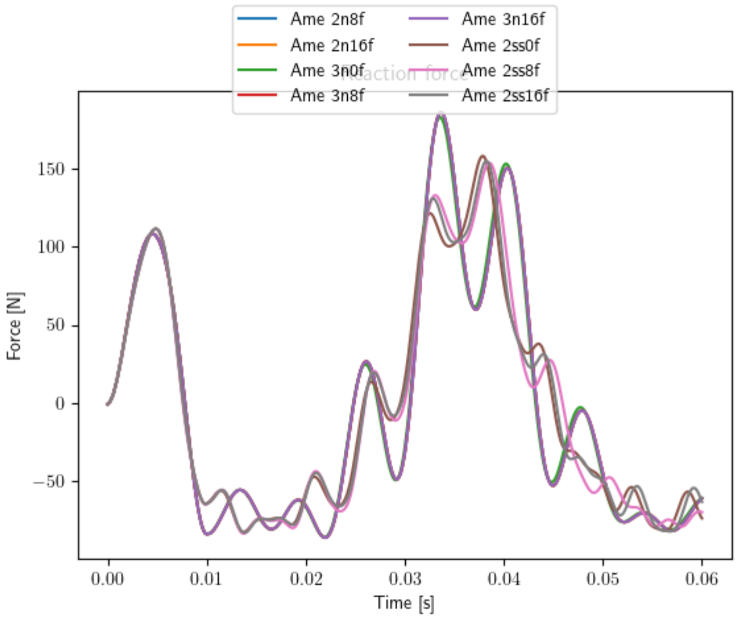


Figure 6.12: Reaction forces at the top of the crank in X direction for all ROMs except the ROM with 2 nodes without eigenmodes in Amesim

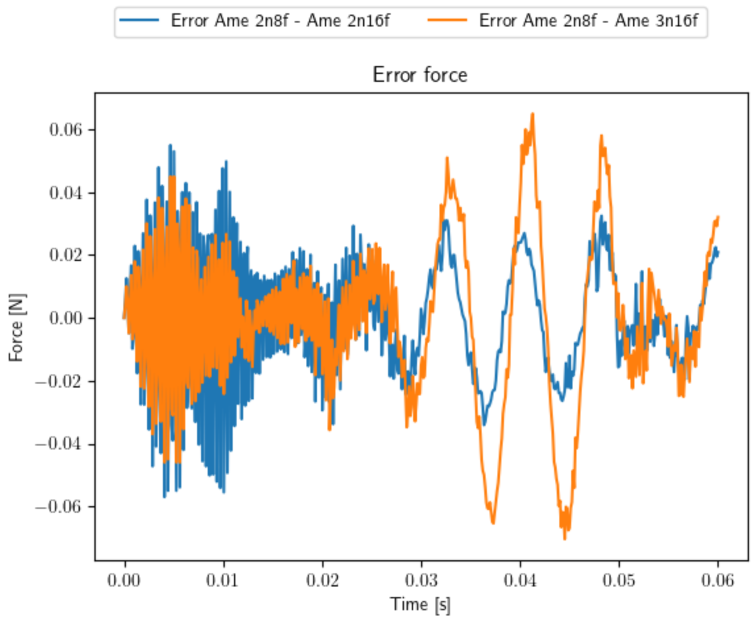


Figure 6.13: Difference in reaction forces as a result of adding eigenmodes in Amesim

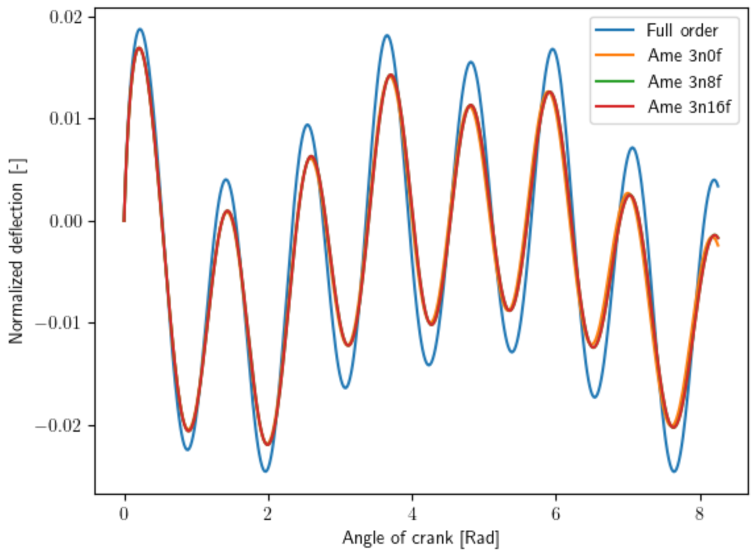


Figure 6.14: Deflection for the full order model and ROM with 3 nodes with 0, 8 and 16 eigenmodes

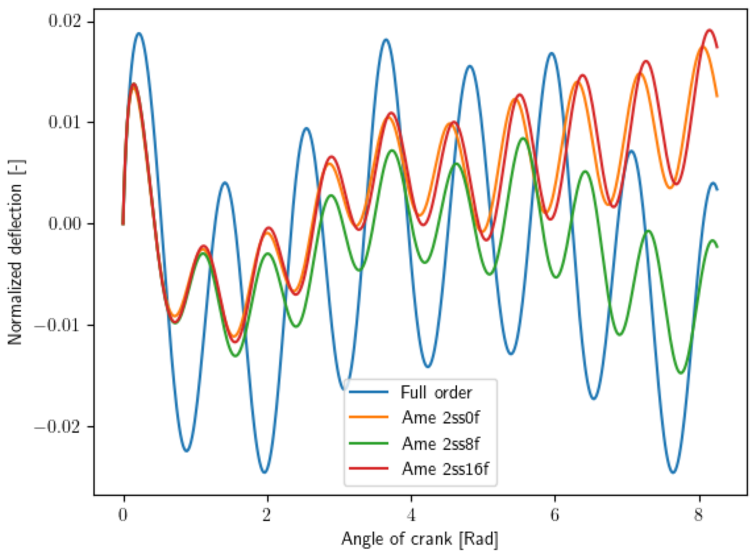


Figure 6.15: Deflection for the full order model and ROM with 2 substructures with 0, 8 and 16 eigenmodes

7 Conclusion & Recommendations

Creating a transient simulation of a ROM has proven more interesting than expected. The way Abaqus determines the location of the computational frames does not work well when a ROM has few nodes. There is currently no way to overwrite, or verify, the location of the computational frame. Adding dynamic eigen modes does improve the accuracy of the results somewhat but has hardly any effect on the computational performance. This implies that, in this case, the size of the system matrix is not the limiting factor for the performance of the solver.

Running the same models in Amesim results in better accuracy, even with less dynamic eigen modes. The computational effect of adding modes is however significant. This seems to indicate that the system matrix of the flexible body is the main contributor of the computational effort. For both Abaqus and Amesim the subdivision of the flexible body into two elements did not positively influence the accuracy of the simulation. When also accounting for the additional work of splitting the original model this approach is not advised.

Apart from the positive results from the Amesim simulations, there are still some downsides of using Amesim. The constraint components can, dependent on their spacial orientation, greatly slow down the simulation. There are incompatibilities that, when encountered, do not result in an (explicit) error message. This resulted in a lot of trial and error to produce working simulations. However these downsides could probably be addressed in future versions.

7.1 Recommendations

Now that there is a working example of a 3D flexible body simulation in Amesim it will be interesting to further explore the system modeling aspects of Amesim. Important is to investigate how the addition of the 1D mechanical elements to simulate imperfections influence the computational performance.

The realtime functionality of Amesim is also be an interesting topic for further research. Looking at the difference in performance between Abaqus and Amesim, there might be a lot to gain performance wise when using a different solver in Amesim.

Bibliography

[1] Sir Isaac Newton. *Philosophiæ Naturalis Principia Mathematica tomus primus: De motu corporum.* MDCCLX (1760).

[2] Przemyslaw A. Lasota, Gregory F. Rossano, and Julie A. Shah. Toward safe close-proximity human-robot interaction with standard industrial robots. volume 2014-January, pages 339–344. IEEE Computer Society, 2014. doi: 10.1109/CoASE.2014.6899348.

[3] Changliu Liu and Masayoshi Tomizuka. Algorithmic safety measures for intelligent industrial co-robots. volume 2016-June, pages 3095–3102. Institute of Electrical and Electronics Engineers Inc., 6 2016. ISBN 9781467380263. doi: 10.1109/ICRA.2016.7487476.

[4] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33:1292–1312, 12 2017. ISSN 15523098. doi: 10.1109/TRO.2017.2723903.

[5] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms, 2018. URL `https://hal.laas.fr/hal-01790971v2`.

[6] IMOCO4.E Consortium. Intelligent Motion Control under Industry4.E - IMOCO4.E, January 2023. `https://www.imoco4e.eu/`.

[7] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29: 36–52, 5 2020. ISSN 17555817. doi: 10.1016/j.cirpj.2020.02.002.

[8] P. Aivaliotis, K. Georgoulias, Z. Arkouli, and S. Makris. Methodology for enabling digital twin using advanced physics-based modelling in predictive maintenance. volume 81, pages 417–422. Elsevier B.V., 2019. doi: 10.1016/j.procir.2019.03.072.

[9] Philips. Allura Xper FD10 X-ray system, May 2023. `https://www.philips.nl/healthcare/product/HC722022CA/allura-xper-fd10-cardiovascular-x-ray-system`.

[10] Marcel Ellenbroek and Jurnan Schilder. On the use of absolute interface coordinates in the floating frame of reference formulation for flexible multibody dynamics. *Multibody System Dynamics*, 43: 193–208, 7 2018. ISSN 1573272X. doi: 10.1007/s11044-017-9606-3.

[11] Hans M Hilber, Thomas J R Hughes, and Robert L Taylor. Improved numerical dissipation for time integration algorithms in structural dynamics, 1977.

[12] Robert J Guyan. Reduction of stiffness and mass matrices. *Aeronautics and Astronautics*, 3:380, 1965. doi: 10.2514/3.2874ï. URL `https://hal.archives-ouvertes.fr/hal-01711552`.

[13] Abaqus Theory Guide. In *Abaqus 2016 Online Documentation*. Dassault Systèmes, 2015.

[14] L R Petzold. Description of dassl: a differential/algebraic system solver. 9 1982. URL `https://www.osti.gov/biblio/5882821`.

[15] K.L. van Voorthuizen, J.P. Schilder, and M.I. Abdul Rasheed. A comparison of reference conditions in floating frame formulations that use absolute interface coordinates. In *Proceedings of the 11th ECCOMAS Thematic Conference on MULTIBODY DYNAMICS*, 2023. (Expected publication august 2023).

[16] Jr. William D. Callister. *Materials science and engineering: an introduction 7th ed.* John Wiley & Sons, Inc., 2007.

[17] Christian Andersson, Claus Führer, and Johan Åkesson. Assimulo: A unified framework for {ODE} solvers. *Mathematics and Computers in Simulation*, 116(0):26 – 43, 2015. ISSN 0378-4754. doi: http://dx.doi.org/10.1016/j.matcom.2015.04.007.

[18] Jian S. Dai. Euler-rodrigues formula variations, quaternion conjugation and intrinsic connections. *Mechanism and Machine Theory*, 92:144–152, 5 2015. ISSN 0094114X. doi: 10.1016/j.mechmachtheory.2015.03.004.

[19] Ronald F Boisvert and Roldan Pozo. The matrix market exchange formats: Initial design, 1997. URL `https://www.researchgate.net/publication/2630533`.

A Eigenmodes slider crank

Mode	Mode description	Frequency [Hz]									
		Full order	2 nodes			3 nodes			2 substructures		
	# of modes in ROM→		0	8	16	0	8	16	0	8	16
	Total DoF →	11295	12	20	28	18	26	34	18	34	50
7	1st bending	299.157	358.75	299.29	299.17	299.85	299.18	299.16	299.85	299.16	299.16
8	1st bending	299.157	358.75	299.29	299.17	299.85	299.18	299.16	299.85	299.16	299.16
9	2nd bending	821.963	1221.18	822.77	822.21	935.78	822.36	822.03	935.78	822.03	821.97
10	2nd bending	821.964	1221.18	822.77	822.21	935.78	822.36	822.03	935.78	822.03	821.98
11	3rd bending	1608.54		1615.75	1609.41	2334.52	1621.25	1609.83	2334.52	1609.83	1608.65
12	3rd bending	1608.54		1615.75	1609.42	2334.52	1621.25	1609.83	2334.52	1609.83	1608.84
13	4th bending	2646.48		2657.63	2651.01	3723.73	2675.49	2649.04	3723.73	2649.04	2646.76
14	4th bending	2646.48		2657.63	2651.01	3723.73	2675.49	2649.04	3723.73	2649.04	2647.09
15	5th bending	3939.47		7692.37	3946.94		4040.37	3946.87		3946.87	3941.56
16	5th bending	3939.47		7692.37	3946.94		4040.37	3946.87		3946.87	3941.56
17	1st torsion	5213.13	5748.29	5748.29	5748.29	5748.29	5748.29	5748.29	5748.29	5748.29	5218.49
18	6th bending	5467.12		11 177.7	5487.50		7947.98	5476.06		5476.06	5469.77
19	6th bending	5467.12		11 177.7	5487.57		7947.98	5476.06		5476.06	5470.07
20	7th bending	7241.12			7267.02		13 874	7334.50		7334.50	7248.32
21	7th bending	7241.13			7267.04		13 874	7334.50		7334.50	7261.27
22	1st longitudinal	8411.34	9269.57	9269.57	9269.57	9269.57	9269.57	9269.57	9269.57	9269.57	8448.77
23	8th bending	9227.56			20 124.57		16 212.7	9342.92		9342.94	9235.02
24	8th bending	9227.56			20 126.65		16 212.7	9342.92		9342.94	9247.86
25	2nd torsion	10 426.3			11 977.70	11 496.6	11 496.6	11 496.60	11 496.6	11 496.6	10 472.81

Table A.1: Eigen frequencies of the various ROMs

Mode number	Mode description	Frequency [Hz]	Error as percentage of frequency [-]								
		Full order	2 nodes			3 nodes			2 substructures		
			0	8	16	0	8	16	0	8	16
	Total DoF →	11295	12	20	28	18	26	34	18	34	50
7	1st bending	299.157	19.92	0.04	0.00	0.23	0.01	0.00	0.23	0.00	0.00
8	1st bending	299.157	19.92	0.04	0.00	0.23	0.01	0.00	0.23	0.00	0.00
9	2nd bending	821.963	48.57	0.10	0.03	13.85	0.05	0.01	13.85	0.01	0.00
10	2nd bending	821.964	48.57	0.10	0.03	13.85	0.05	0.01	13.85	0.01	0.00
11	3rd bending	1608.54		0.45	0.05	45.13	0.79	0.08	45.13	0.08	0.01
12	3rd bending	1608.54		0.45	0.05	45.13	0.79	0.08	45.13	0.08	0.02
13	4th bending	2646.48		0.42	0.17	40.70	1.10	0.10	40.70	0.10	0.01
14	4th bending	2646.48		0.42	0.17	40.70	1.10	0.10	40.70	0.10	0.02
15	5th bending	3939.47		95.26	0.19		2.56	0.19		0.19	0.05
16	5th bending	3939.47		95.26	0.19		2.56	0.19		0.19	0.05
17	1st torsion	5213.13	10.27	10.27	10.27	10.27	10.27	10.27	10.27	10.27	0.10
18	6th bending	5467.12		104.45	0.37		45.38	0.16		0.16	0.05
19	6th bending	5467.12		104.45	0.37		45.38	0.16		0.16	0.05
20	7th bending	7241.12			0.36		91.60	1.29		1.29	0.10
21	7th bending	7241.13			0.36		91.60	1.29		1.29	0.28
22	1st longitudinal	8411.34	10.20	10.20	10.20	10.20	10.20	10.20	10.20	10.20	0.45
23	8th bending	9227.56			118.09		75.70	1.25		1.25	0.08
24	8th bending	9227.56			118.11		75.70	1.25		1.25	0.22
25	2nd torsion	10 426.3			14.88	10.27	10.27	10.27	10.27	10.27	0.45

Table A.2: Error of eigen frequency as percentage of full order model for the various ROMs

A.1 Free-free mode shapes

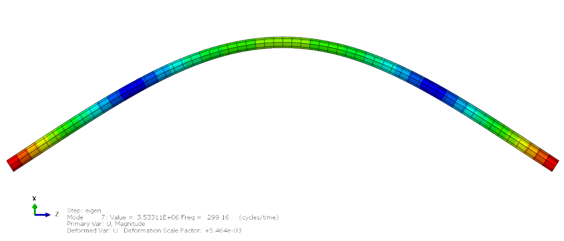

x
z
Step: eigen
Mode 7: Value = 3.53311E+06 Freq = 299.16 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.464e-03

Figure A.1: 1st bending mode (mode 7 & 8)

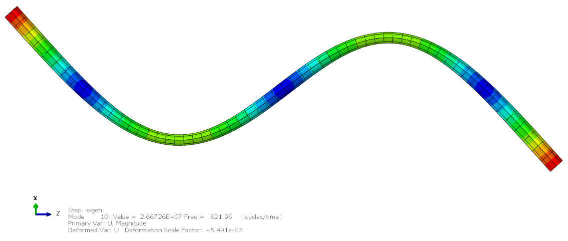

x
z
Step: eigen
Mode 10: Value = 2.66726E+07 Freq = 821.96 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.491e-03

Figure A.2: 2nd bending mode (mode 9 & 10)

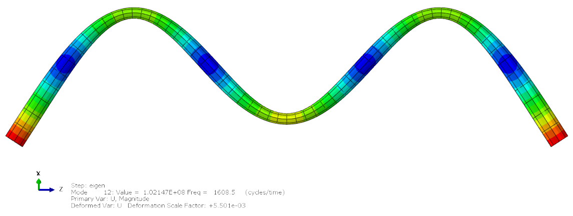

x
z
Step: eigen
Mode 12: Value = 1.02147E+08 Freq = 1608.5 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.501e-03

Figure A.3: 3rd bending mode (mode 11 & 12)

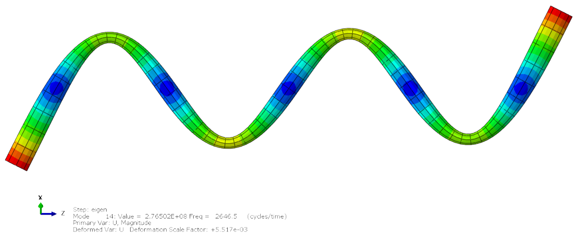

x
z
Step: eigen
Mode 14: Value = 2.76502E+08 Freq = 2646.5 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.517e-03

Figure A.4: 4th bending mode (mode 13 & 14)

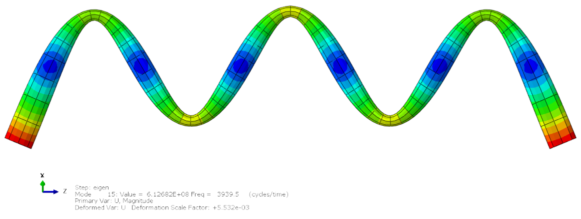

x
z
Step: eigen
Mode 15: Value = 6.12682E+08 Freq = 3939.5 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.532e-03

Figure A.5: 5th bending mode (mode 15 & 16)

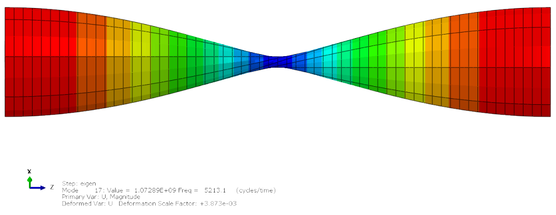

x
z
Step: eigen
Mode 17: Value = 1.07289E+09 Freq = 5213.1 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +3.873e-03

Figure A.6: 1st torsion mode (mode 17)

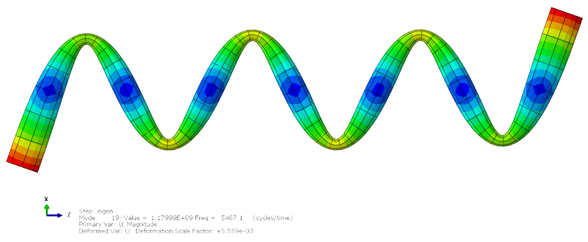

x
z
Step: eigen
Mode 19: Value = 1.17999E+09 Freq = 5467.1 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.559e-03

Figure A.7: 6th bending mode (mode 18 & 19)

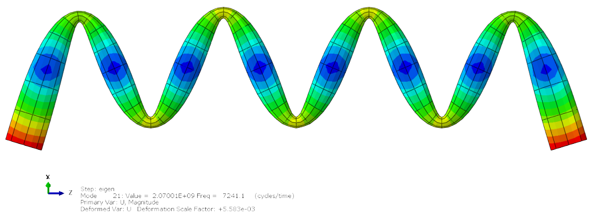

x
z
Step: eigen
Mode 21: Value = 2.07001E+09 Freq = 7241.1 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.583e-03

Figure A.8: 7th bending mode (mode 20 & 21)

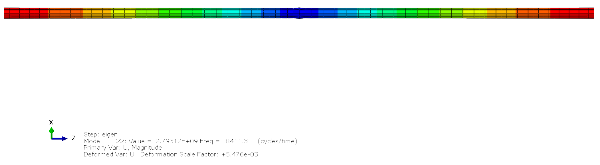

x
z
Step: eigen
Mode 22: Value = 2.79312E+09 Freq = 8411.3 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.476e-03

Figure A.9: 1st longitudinal mode (mode 22)

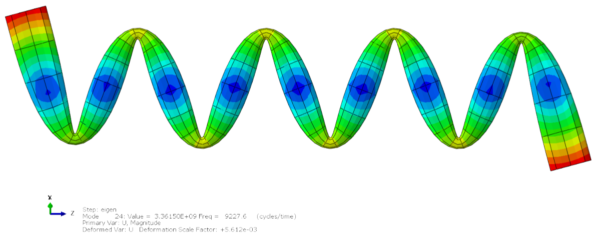

x
z
Step: eigen
Mode 24: Value = 3.36150E+09 Freq = 9227.6 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.612e-03

Figure A.10: 8th bending mode (mode 23 & 24)

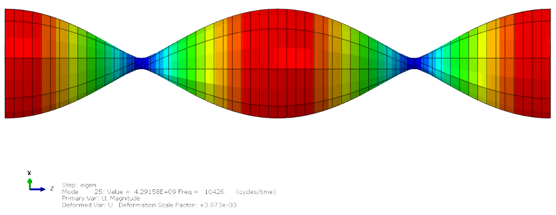
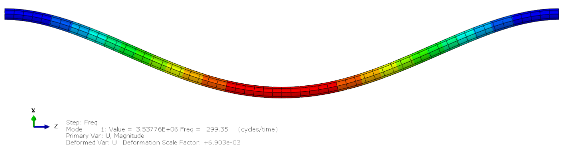

x
z
Step: eigen
Mode 25: Value = 4.29158E+09 Freq = 10426. (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +3.873e-03

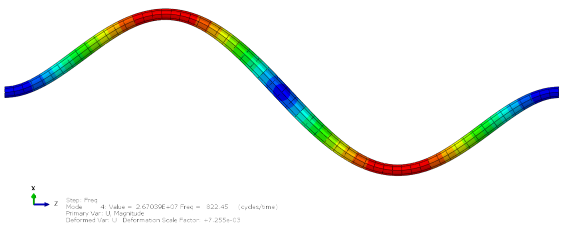
Figure A.11: 2nd torsion mode (mode 25)

A.2 Fixed-fixed mode shapes



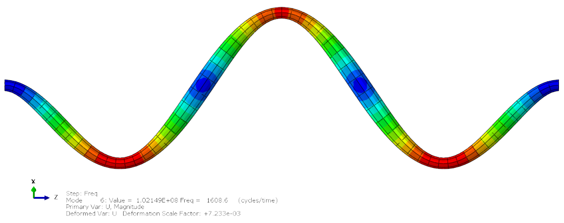
```
Step: Freq
Mode        1: Value =  3.53776E+06 Freq =   299.35    (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U   Deformation Scale Factor: +6.903e-03
```

Figure A.12: 1st bending mode (mode 1 & 2)



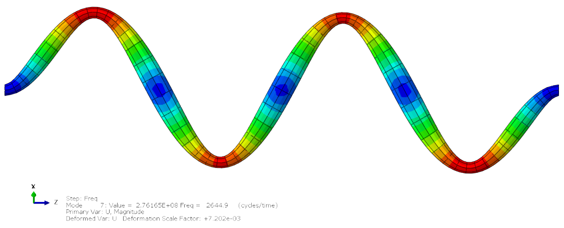
```
Step: Freq
Mode        4: Value =  2.67039E+07 Freq =   822.45    (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U   Deformation Scale Factor: +7.255e-03
```

Figure A.13: 2nd bending mode (mode 3 & 4)



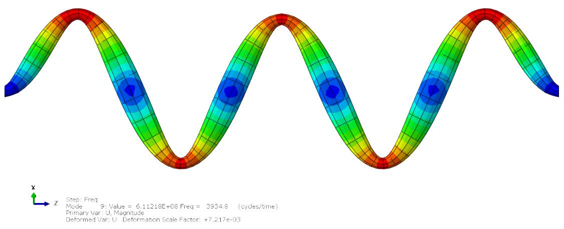
```
Step: Freq
Mode        6: Value =  1.02149E+08 Freq =   1608.6    (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U   Deformation Scale Factor: +7.233e-03
```

Figure A.14: 3rd bending mode (mode 5 & 6)



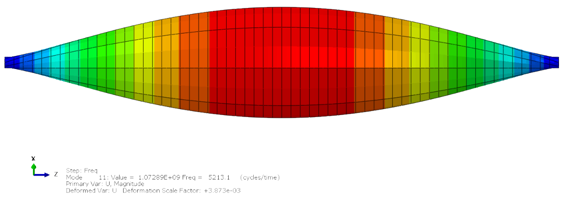
```
Step: Freq
Mode        7: Value =  2.76165E+08 Freq =   2644.9    (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U   Deformation Scale Factor: +7.202e-03
```

Figure A.15: 4th bending mode (mode 7 & 8)



```
Step: Freq
Mode        9: Value =  6.11218E+08 Freq =   3934.8    (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U   Deformation Scale Factor: +7.217e-03
```

Figure A.16: 5th bending mode (mode 9 & 10)



```
Step: Freq
Mode       11: Value =  1.07289E+09 Freq =   5213.1    (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U   Deformation Scale Factor: +3.873e-03
```

Figure A.17: 1st torsion mode (mode 11)

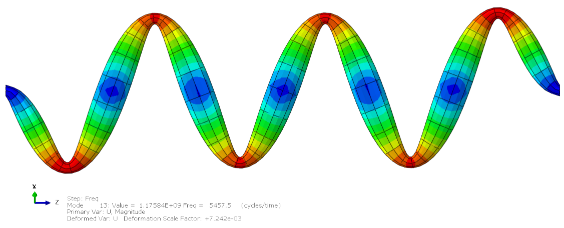

x
z
Step: Freq
Mode 13: Value = 1.17584E+09 Freq = 5457.5 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +7.242e-03

Figure A.18: 6th bending mode (mode 12 & 13)

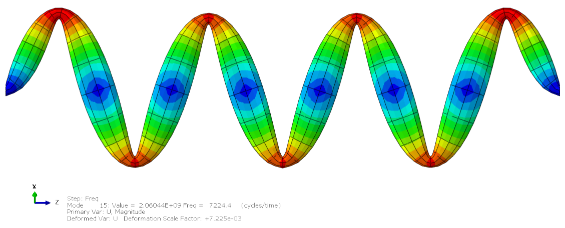

x
z
Step: Freq
Mode 15: Value = 2.06044E+09 Freq = 7224.4 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +7.225e-03

Figure A.19: 7th bending mode (mode 14 & 15)

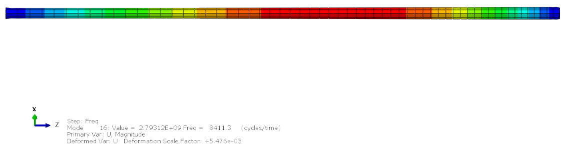

x
z
Step: Freq
Mode 16: Value = 2.79312E+09 Freq = 8411.3 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.476e-03

Figure A.20: 1st longitudinal mode (mode 16)

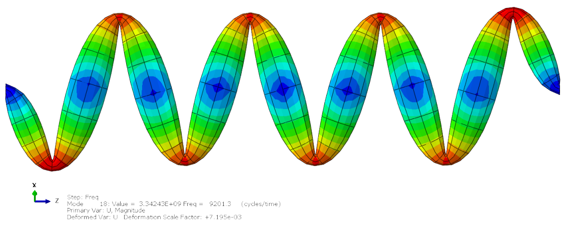

x
z
Step: Freq
Mode 18: Value = 3.34243E+09 Freq = 9201.3 (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +7.195e-03

Figure A.21: 8th bending mode (mode 17 & 18)

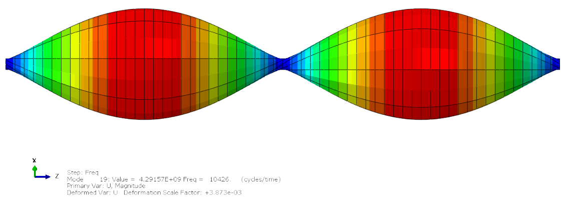

x
z
Step: Freq
Mode 19: Value = 4.29157E+09 Freq = 10426. (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +3.873e-03

Figure A.22: 2nd torsion mode (mode 19)

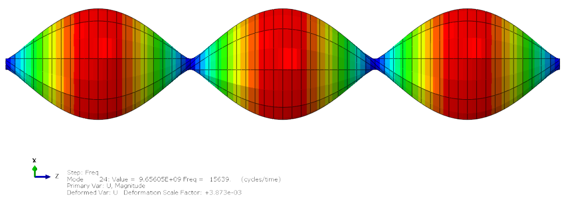

x
z
Step: Freq
Mode 24: Value = 9.65605E+09 Freq = 15639. (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +3.873e-03

Figure A.23: 3rd torsion mode (mode 24)

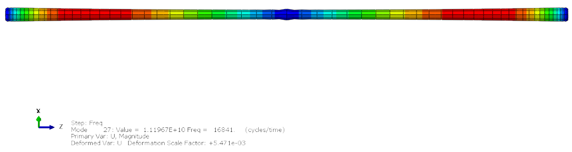

x
z
Step: Freq
Mode 27: Value = 1.11967E+10 Freq = 16841. (cycles/time)
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +5.471e-03

Figure A.24: 2nd longitudinal mode (mode 27)

B Abaqus smooth step

When prescribing a boundary condition for a dynamic step in Abaqus two parameters are needed: the value of the BC at the end of the step and a function that describes the behavior of the value in time. This function of time is called an *Amplitude* in Abaqus. One of these build-in amplitude functions is the smooth step profile. The smooth step function is a polynomial (equation (B.1)) where the first and second derivatives are zero at the start and end of the domain. The smooth step is applied to the angular displacement of the center point of the slider crank.

To prevent high frequency exitations the build-in *Smooth step* amplitude of Abaqus was used. A smooth step between amplitude A_i and A_{i+1} is defined by equation (B.1). The smooth step, and its derivatives are visualized in figure B.1.

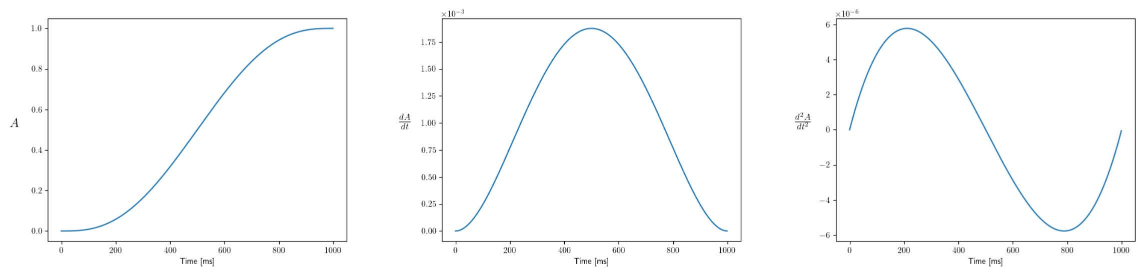
$$\begin{aligned} a = A_i + (A_{i+1} - A_i)\xi^3(10 - 15\xi + 6\xi^2) \qquad \text{for} \qquad & t_i \le t \le t_{i+1} \\ \text{with} \quad \xi(t) = & \frac{t - t_i}{t_{i+1} - t_i} \end{aligned} \tag{B.1}$$

Since the equation for the smooth step is a polynomial determining its derivatives and antiderivatives is, although lengthy, straightforward. If the smooth step is used to prescribe the angle of a joint, the first and second derivatives of the smooth step function are required to determine the velocities and acceleration as functions of the time. The antiderivatives are required when the acceleration (or velocity) was prescribed with the smooth step function to obtain a function for the angle and the acceleration.

When prescribing the rotational velocity or acceleration multiple smooth step functions are sequentially used to create a motion that is similar to the one when the angle is prescribed. When determining the profile for angle by integration from a prescribed acceleration profile, the two integration constants that appear are used to assure continuity of the angle in the domain.

To determine the expressions for the derivatives and antiderivatives the expression for ξ was first substituted into the expression for the smooth step. The symbolic toolbox from MATLAB was used to perform the derivation and integration steps. For the step as visualized in figure B.1a the terms A_i & t_i are zero and the derivatives and antiderivatives are given in equation (B.2)[1]. Since the full expressions are very lengthy they are not included in this report.

[1] The integration constants are both zero so they drop out



(a) Amplitude of the smooth step function (b) 1st derivative of the smooth step function (c) 2nd derivative of the smooth step function

Figure B.1: The Abaqus smooth step function from 0 to 1 in 1 second

$$
\begin{aligned}
\ddot{a} &= 60tA_{i+1}\frac{2t^2 - 3t_{i+1}t + t_{i+1}^2}{t_{i+1}^5} \\
\dot{a} &= 30A_{i+1}t^2\frac{(t - t_{i+1})^2}{t_{i+1}^5} \\
a &= A_{i+1}t^3\frac{6t^2 - 15t_{i+1}t + 10t_{i+1}^2}{t_{i+1}^5} \\
\int_0^t a\,dt &= A_{i+1}t^4\frac{2t^2 - 6t_{i+1}t + 5t_{i+1}^2}{2t_{i+1}^5} \\
\int_0^t \left(\int_0^t a\,dt\right)\,dt &= A_{i+1}t^5\frac{2t^2 - 7t_{i+1}t + 7t_{i+1}^2}{14t_{i+1}^5}
\end{aligned}
\tag{B.2}
$$

C Center deflection

The method used by [10] to determine a scalar value for the center deflection is shown in this appendix. The idea is to construct a rotation matrix to transform the global deformations to deformations local to the connector. For this a rotation vector and angle are determined. The rotation vector $\boldsymbol{v}$ is chosen to be perpendicular to the plane defined by the x-axis and the line AB, and is normalized. Assumed is that there is no rotation along the x-axis. The rotation angle is the angle between these lines (in the undeformed state). The resulting coordinate transformation can be seen in figure C.1a.
The unit vector $\boldsymbol{n}$ along the (undeformed) flexible connector.

$$\boldsymbol{L} = \begin{Bmatrix} x_B \\ -y_A \\ -z_A \end{Bmatrix}, \qquad \boldsymbol{n} = \frac{\boldsymbol{L}}{|\boldsymbol{L}|} \tag{C.1}$$

The rotation vector $\boldsymbol{v}$ becomes

$$\bar{\boldsymbol{v}} = \begin{Bmatrix} 0 \\ -n_3 \\ n_2 \end{Bmatrix}, \qquad \boldsymbol{v} = \frac{\bar{\boldsymbol{v}}}{|\bar{\boldsymbol{v}}|} \tag{C.2}$$

The expression for the rotation angles

$$\sin\phi = -\frac{\sqrt{y_A^2 + z_A^2}}{L_{rod}}, \qquad \cos\phi = \frac{x_B}{L_{rod}} \tag{C.3}$$

The rotation matrix is calculated using the following formula[18]

$$\boldsymbol{R} = \mathbf{1} + \tilde{\boldsymbol{v}}\sin\phi + \tilde{\boldsymbol{v}}\tilde{\boldsymbol{v}}(1 - \cos\phi) \tag{C.4}$$

where

$$\tilde{\boldsymbol{v}} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{C.5}$$

The deflection is determined

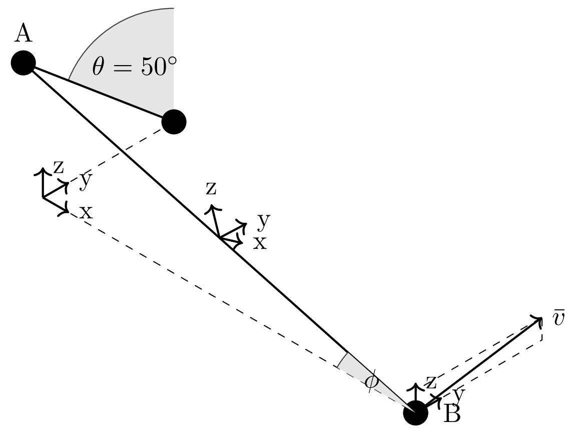
$$\boldsymbol{d} = \boldsymbol{R}^{\intercal}(\boldsymbol{x}_{mid} - \boldsymbol{x}_{mid,undeformed}) \tag{C.6}$$

with

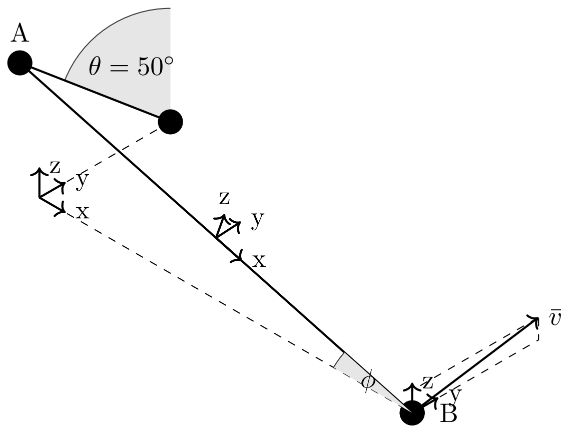
$$\boldsymbol{x}_{mid,undeformed} = \frac{1}{2}\begin{Bmatrix} x_B \\ y_A \\ z_A \end{Bmatrix} \tag{C.7}$$

The normalized deflection that can be seen in the various deflection plots is obtained by taking the y value of the deflection vector and dividing by the length of the rod.

In the scripts provided by Karlijn, the deflection is obtained using equation (C.6). Plotting this coordinate transformation shows an illogical result. Plotting the coordinate transformation with the inverse of the rotation matrix seems to provide better results, see figure C.1b. Since the goal of this deflection is purely to compare the results obtained with Abaqus and Amesim to results from the python code using the Floating Frame Formulation equation (C.6) is used in this research.



(a) The coordinate transformation for a rotation angle of 50° as calculated by the rotation matrix from equation (C.4)



(b) The coordinate transformation for a rotation angle of 50° as calculated by the inverse of the rotation matrix from equation (C.4)

D Converting an Abaqus substructure to Amesim modal data

By default Abaqus does not export the mass and stiffness matrices during substructure generation in an open format. There is a way to export these values but it is not supported in the Abaqus CAE so the input file for the solver must be edited manually. In the substructure generation step the generation of the mass matrix must also be manually enabled.

```
*SUBSTRUCTURE MATRIX OUTPUT, file name=systemMatrices, mass=YES, OUTPUT
FILE=USER DEFINED, STIFFNESS=YES
```

By adding this line a .mtx file is generated containing the order of the nodes and their exported DoF, the number of nodes corresponding to normal modes, and the sparse representation of the mass and stiffness matrices. The file extension implies a file formatted according to the Matrix Market Exchange Format[19], the file does not comply with this format but the matrices are stored according to the Hermitian format. At Reden a python script was already available to parse the mtx file into a numpy array.

From the scipy linear algebra toolkit the *eigh* function can be used to solve the eigenvalue problem where the stiffness matrix should be the first argument and the mass matrix the second. This function returns the eigenvectors as columns in the matrix

The final steps are to trim the first 6 eigenvalues and corresponding eigenvectors, removing the rows corresponding to the normal modes and saving the transpose of the result. Amesim expects the eigenvectors as rows in the matrix.

E Center deflection

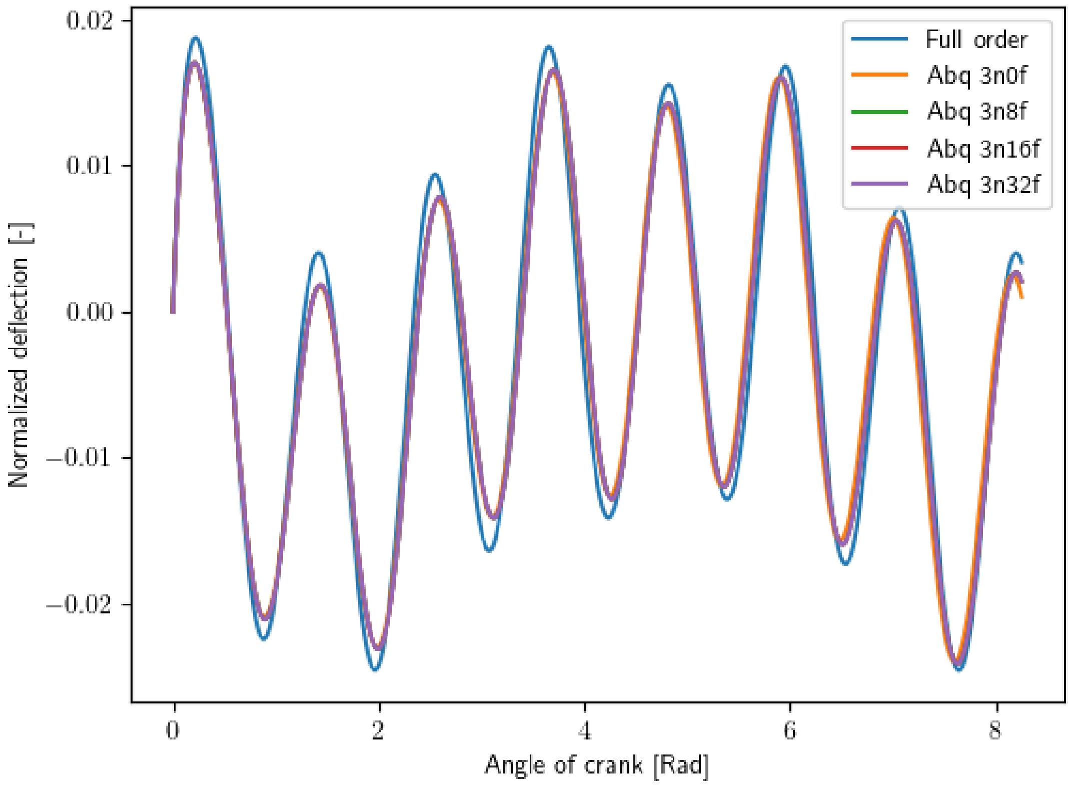


Figure E.1: Center deflection for ROM with 3 nodes

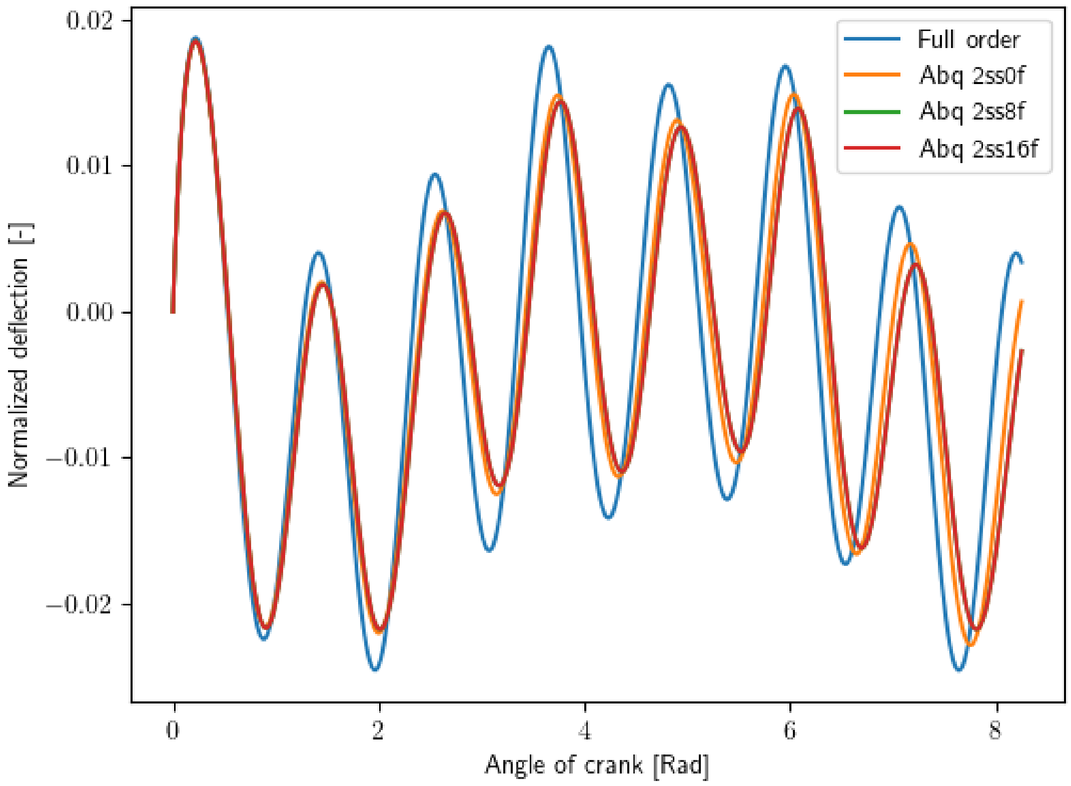


Figure E.2: Center deflection for ROM with 2 substructures