Master thesis

Uncertainty quantification in flexible multibody dynamics

By:
Karlijn van Voorthuizen
s1600907

Supervisors:
Prof. Dr.-Ing. Bojana Rosic
Dr. ir. Jurnan Schilder

Faculty of Engineering Technology
Department of Mechanics of Solids, Surfaces & Systems

2019-2020

UNIVERSITY OF TWENTE.

1 Abstract

Simulations of flexible dynamic systems are generally performed under the assumption that the input variables, such as the geometric and material properties, are exactly known. In actuality, a number of geometric and material properties are uncertain. To determine the influence of the uncertain parameters on the output of the model, one can sample the distributions of each uncertain parameter a large number of times, evaluate the deterministic dynamic model for each sample and use the obtained samples of the output parameter to estimate its distribution. However, simulations of flexible dynamic systems typically are expensive. Evaluating the simulation for each sample is therefore not feasible. By reduction of the model, the evaluation time might be reduced to such extend that the large number of evaluations can be performed within an appropriate time frame. The deterministic model can be reduced through physics-based methods or by a data-driven approach to reduce the number of unknown variables in the model. Alternatively, the stochastic model can be reduced by discretization of the stochastic space to obtain a metamodel which approximates the original system but consists of polynomials which are inexpensive to evaluate. This report shows a successful application of different data-driven reduction techniques on deterministic and stochastic flexible dynamic systems.

Contents

2 Introduction

Flexible multibody dynamics (FMBD) considers the equations of motion of complex dynamic systems of which the bodies can deform flexibly during movement [1, 2]. The type of structures on which flexible multibody dynamics is applied is diverse, but some typical examples are construction cranes and amusement park attractions such as roller coasters. By simulation of such structures, knowledge of its motion, present vibrations and occurring forces is obtained, but running the simulation is typically time consuming. Additionally, the knowledge has been obtained under the assumption that the input parameters have a specific setting, while often several aspects of the modelled systems are not known exactly. Think for example of the dimensions of the different bodies but also of the material properties. The question then changes from "what is the outcome of the simulation for specific input parameters?" to "how likely are different outcomes of the simulation to occur?".

A well-known and simple method to attain the distributions of the output parameters is Monte Carlo (MC), which requires the evaluation of the FMBD simulation for a large number of samples of the distributions of each uncertain input parameter [3]. One can use the resulting set of samples of the output parameters to estimate their respective distributions. However, as FMBD simulations are typically expensive, application of MC on the models is unfeasible. To obtain an understanding of the influence of random variables on a flexible multibody dynamic system within an acceptable time frame, reduction must be achieved.

Because the application of MC requires the evaluation of the deterministic model, significant reduction of the deterministic model might make MC feasible. Reduction of the deterministic system can be achieved by discretization of the flexible coordinates using free boundary modes or by applying the Craig Bampton method [1, 2]. Both are physics-based methods which reduce the number of flexible coordinates required to simulate the FMBD system and are often applied in flexible multibody problems. Alternatively, the space might be discretized by the data-driven technique proper orthogonal decomposition (POD) [4, 5]. POD requires a snapshot matrix containing the output variables of the FMBD simulation at successive time instances, on which singular value decomposition is applied to find the dominant relations between the output variables over time. The dominant relations form the set of the POD modes used to reduce the model. One can apply the physics-based and data-driven methods to determine whether POD can achieve a similar or superior reduction compared to the physics-based methods while the error between the reduced and unreduced simulations is similar.

By introducing uncertainties into the system, the stochastic model is obtained. The stochastic model can be reduced by discretization of the stochastic space through polynomial chaos expansion (PCE) [6, 7, 8]. The expansions consist of the set of orthonormal polynomials and scaling factors. The scaling factors are tuned using of a training data set such that the expansions approximate the relations between the input and output variables. The expansions are less expansive to evaluate compared to the original stochastic model because the polynomials can be evaluated in seconds. Whether expansions can be obtained which approximate a dynamic model closely while reducing the simulation time to such extent that Monte Carlo is feasible will be investigated.

The Craig Bampton modes, which are used to achieve discretization in space, are dependent on the material properties of the system. In case of uncertain material properties, the stochastic space of the Craig Bampton modes can be discretized as well. One possibility to achieve the discretization is to apply POD in the stochastic space, which allows one to investigate the relation between POD modes and Craig Bampton modes. Alternatively, PC expansions of the modes might be obtained by stochastic Galerkin, in which the stochastic space is discretized using polynomials but the scaling factors are not obtained using a training data set [9, 10]. Instead, PC expansions are formulated for each variable in the definition of the Craig Bampton modes and used to obtain a relation between the known scaling factors of the expansions of the uncertain input parameters and the unknown scaling factors of the output variables. Whether the discretization through POD or through stochastic Galerkin results is a good approximation of the Craig Bampton modes and whether significant reduction can be achieved through the methods will be explored.

In summation, four data-driven approaches to reduce FMBD models will be examined: the deterministic FMBD model will be reduced by POD, the stochastic FMBD model will be reduced by PCE and the stochastic Craig Bampton modes will be discretized in the stochastic space using POD and stochastic Galerkin.

In the first chapter of this report the model problem will be derived, involving the derivation of the constrained equations of motion, the discretization in space and techniques required for the integration

over time. The second chapter presents three methods to achieve reduction of the deterministic constrained equations of motion (free boundary modes, Craig Bampton and POD), while the following chapter introduces uncertainties to the model and considers the discretization of the stochastic space using PCE. The next chapter considers the reduction of the uncertain space of the Craig Bampton modes through POD and through stochastic Galerkin. Finally, the described reduction techniques are applied to the benchmark problem of the slider crank to investigate the performance of the different techniques. If the reductions can be achieved for the benchmark problem, considering the application of the techniques on more complex systems might be of interest in future research.

3 Model problem

3.1 Derivation of the constrained equations of motion

The motion of flexible multibody dynamic systems is examined by solving the equation of motion:

$$\boldsymbol{M}_{FE}\ddot{\boldsymbol{q}}_n + \boldsymbol{K}_{FE}\boldsymbol{q}_n = \boldsymbol{F} \tag{3.1}$$

Equation (3.1) represents the discretized version of the dynamic equation for the system in consideration. Therefore, $\boldsymbol{q}_n$ denotes a vector of nodal coordinates expressed in a local frame after the system is discretized by finite elements (FE). Correspondingly, $\ddot{\boldsymbol{q}}_n$ is a vector of the accelerations, $\boldsymbol{M}$ is the FE mass matrix, $\boldsymbol{K}$ is the FE stiffness matrix and $\boldsymbol{F}$ is the resultant force vector. The equation of motion must be solved while ensuring the constraints imposed upon the system are satisfied:

$$\boldsymbol{\Phi}(\boldsymbol{q}_n, t) = \boldsymbol{0} \tag{3.2}$$

where $\boldsymbol{\Phi}$ is a vector of the constraints and t is time [1].

Let be given a global frame with the origin in O as schematically represented in Figure 3.1 and the corresponding floating frame in the center of mass (point 1) of the body as shown in Figure 3.1 as well. One can pass between frames by rotation and translation of the original frame. The translation is defined as the positions of the origins of the frames, while the rotation is defined by the rotation of the body to a different orientation.

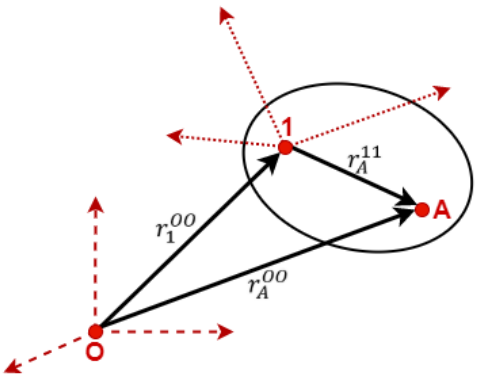


Figure 3.1: The position vector of point A on some body expressed using the global frame O and the floating frame 1.

To move from one coordinate system to another, one may use the transformation, i.e. the rotation matrix such that

$$\begin{bmatrix} x^O \\ y^O \\ z^O \end{bmatrix} = \boldsymbol{R}_1^O \begin{bmatrix} x^1 \\ y^1 \\ z^1 \end{bmatrix} \tag{3.3}$$

holds, where x^O, y^O and z^O are the elements of a position vector expressed in frame O, x^1, y^1 and z^1 are the elements of the same position vector expressed in frame 1 as schematically shown in Figure 3.2 and $\boldsymbol{R}_1^O$ is the rotation matrix of frame 1 with respect to frame O [1, 2, 11].

The position vector of the origin of the floating frame with respect to the global frame defines the rigid movement of the body, while the position vector of a point on the body with respect to the floating frame represents the flexible movement [1, 2]. The position of a point A on the body can be described as the sum of the rigid movement and the position vector of point A with respect to the floating frame:

$$\boldsymbol{r}_A^{OO} = \boldsymbol{r}_1^{OO} + \boldsymbol{r}_A^{O1} \tag{3.4}$$

where $\boldsymbol{r}_A^{OO}$ is the vector defining the position of point A (the subscript) with respect to point O (the second superscript) expressed in frame O (the first superscript), $\boldsymbol{r}_1^{OO}$ is the vector representing the position of point

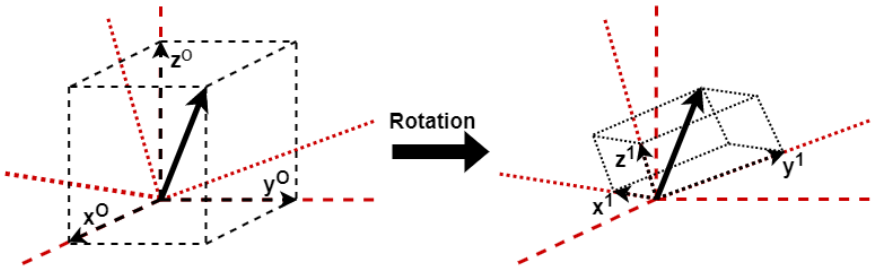


Figure 3.2: Schematic illustration of the rotation between the global frame O and the floating frame 1.

1 with respect to point O expressed in frame O and $\boldsymbol{r}_A^{O1}$ is the vector describing the position of point A with respect to point 1 expressed in frame O, as schematically shown in Figure 3.1. The rigid motion of the body is described by $\boldsymbol{r}_1^{OO}$ while $\boldsymbol{r}_A^{O1}$ represents the flexible motion, meaning the rigid and flexible motion are decoupled. Using the rotation matrix, the flexible behaviour can be written in terms of the floating frame:

$$\boldsymbol{r}_A^{OO} = \boldsymbol{r}_1^{OO} + \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} \tag{3.5}$$

Application of the floating frame definition to the equation of motion yields:

$$\boldsymbol{M}_{FE}^1 \ddot{\boldsymbol{q}}_n^{1O} + \boldsymbol{K}_{FE}^1 \boldsymbol{q}_n^{11} = \boldsymbol{F}^1 \tag{3.6}$$

where $\boldsymbol{M}_{FE}^1$ is the finite element mass matrix expressed in the floating frame, $\boldsymbol{K}_{FE}^1$ is the finite element stiffness matrix expressed in the floating frame and $\boldsymbol{F}^1$ is the resultant force vector expressed in the floating frame. The vector of accelerations contains both the accelerations of the rigid and flexible movement of the nodes, while the vector of coordinates contains only the coordinates of the flexible movement of the nodes since multiplication with the stiffness matrix regards the elastic forces. By use of rotation matrices (3.6) can be expressed with respect to the global frame, resulting in the non-linear equation of motion:

$$\boldsymbol{R}_1^O \boldsymbol{M}_{FE}^1 \boldsymbol{R}_O^1 \ddot{\boldsymbol{q}}_n^{OO} + \boldsymbol{R}_1^O \boldsymbol{K}_{FE}^1 \boldsymbol{q}_n^{11} = \boldsymbol{R}_1^O \boldsymbol{F}^1 \tag{3.7}$$

One can expand the equation of motion by writing the accelerations in terms of the floating frame using definitions similar to (3.5). The acceleration vector contains both the translational accelerations and the angular accelerations. If point A is one node in the mesh, the translational accelerations of point A are found by first defining its position:

$$\boldsymbol{r}_A^{OO} = \boldsymbol{r}_1^{OO} + \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} \tag{3.8}$$

and taking the first derivative of the position with respect to time to find the translational velocity vector of point A:

$$\dot{\boldsymbol{r}}_A^{OO} = \dot{\boldsymbol{r}}_1^{OO} + \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} + \boldsymbol{R}_1^O \dot{\boldsymbol{r}}_A^{11} \tag{3.9}$$

where $\boldsymbol{\omega}$ is the vector of angular velocities and $\tilde{\boldsymbol{\omega}}$ is the skew symmetric matrix of the vector of angular velocities. The second derivative of (3.8) with respect to time gives the formulation of the translational acceleration vector of point A:

$$\ddot{\boldsymbol{r}}_A^{OO} = \ddot{\boldsymbol{r}}_1^{OO} + \dot{\tilde{\boldsymbol{\omega}}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} + \tilde{\boldsymbol{\omega}}_1^{OO} \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} + 2\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \dot{\boldsymbol{r}}_A^{11} + \boldsymbol{R}_1^O \ddot{\boldsymbol{r}}_A^{11} \tag{3.10}$$

where $\dot{\boldsymbol{\omega}}$ is the vector of angular accelerations and $\dot{\tilde{\boldsymbol{\omega}}}$ is the skew-symmetric matrix of the vector of angular accelerations. The formulation of the angular accelerations of point A in the floating frame definition is found by first defining the angular velocities:

$$\boldsymbol{\omega}_A^{OO} = \boldsymbol{\omega}_1^{OO} + \boldsymbol{R}_1^O \boldsymbol{\omega}_A^{OO} \tag{3.11}$$

The derivative of the angular velocity vector (3.11) with respect to time results in the vector of angular accelerations of point A:

$$\dot{\boldsymbol{\omega}}_A^{OO} = \dot{\boldsymbol{\omega}}_1^{OO} + \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{\omega}_A^{11} + \boldsymbol{R}_1^O \dot{\boldsymbol{\omega}}_A^{11} \tag{3.12}$$

One can obtain the vector of all accelerations of node A ($\ddot{\boldsymbol{q}}_A^{OO}$) by combining the translational acceleration vector (3.10) and the angular acceleration vector (3.12):

$$\ddot{\boldsymbol{q}}_A^{OO} = \begin{bmatrix} \mathbf{1} & -\boldsymbol{R}_1^O \tilde{\boldsymbol{r}}_A^{11} \boldsymbol{R}_O^1 \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{r}}_1^{OO} \\ \dot{\boldsymbol{\omega}}_1^{OO} \end{bmatrix} + \begin{bmatrix} \boldsymbol{R}_1^O & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_1^O \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{r}}_A^{11} \\ \dot{\boldsymbol{\omega}}_A^{11} \end{bmatrix} + \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{OO} \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} + 2\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \dot{\boldsymbol{r}}_A^{11} \\ \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{\omega}_A^{11} \end{bmatrix} \tag{3.13}$$

where $\tilde{\boldsymbol{r}}_A^{11}$ is the skew-symmetric matrix of the position vector $\boldsymbol{r}_A^{11}$, $\mathbf{1}$ is the identity matrix and $\mathbf{0}$ is a matrix containing zeros. The rotation matrices in the first and last term on the right hand side of (3.13) can be decoupled using the orthogonality property of the rotation matrix:

$$\ddot{\boldsymbol{q}}_A^{OO} = \begin{bmatrix} \boldsymbol{R}_1^O & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_1^O \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\tilde{\boldsymbol{r}}_A^{11} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_O^1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_O^1 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{r}}_1^{OO} \\ \dot{\boldsymbol{\omega}}_1^{OO} \end{bmatrix} + \begin{bmatrix} \boldsymbol{R}_1^O & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_1^O \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{r}}_A^{11} \\ \dot{\boldsymbol{\omega}}_A^{11} \end{bmatrix} + \begin{bmatrix} \boldsymbol{R}_1^O & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_1^O \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{r}_A^{11} + 2\tilde{\boldsymbol{\omega}}_1^{1O} \dot{\boldsymbol{r}}_A^{11} \\ \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{\omega}_A^{11} \end{bmatrix} \tag{3.14}$$

The definition of the accelerations (3.14) can be written compactly by defining

$$\left[\mathbf{R}_1^O\right] = \begin{bmatrix} \boldsymbol{R}_1^O & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_1^O \end{bmatrix} \qquad \left[\mathbf{R}_O^1\right] = \begin{bmatrix} \boldsymbol{R}_O^1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_O^1 \end{bmatrix} \tag{3.15}$$

as well as

$$\left[-\tilde{\mathbf{r}}_A^{11}\right] = \begin{bmatrix} \mathbf{1} & -\tilde{\boldsymbol{r}}_A^{11} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3.16}$$

and

$$\left[\dot{\mathbf{q}}_A^2\right] = \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{r}_A^{11} + 2\tilde{\boldsymbol{\omega}}_1^{1O} \dot{\boldsymbol{r}}_A^{11} \\ \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{\omega}_A^{11} \end{bmatrix} \tag{3.17}$$

which allows the acceleration vector (3.14) to be written as:

$$\ddot{\boldsymbol{q}}_A^{OO} = \left[\mathbf{R}_1^O\right] \left[-\tilde{\mathbf{r}}_A^{11}\right] \left[\mathbf{R}_O^1\right] \ddot{\boldsymbol{q}}_1^{OO} + \left[\mathbf{R}_1^O\right] \ddot{\boldsymbol{q}}_A^{11} + \left[\mathbf{R}_1^O\right] \left[\dot{\mathbf{q}}_A^2\right] \tag{3.18}$$

By extending the acceleration vector (3.18) to include all N nodes in the body, one can write:

$$\ddot{\boldsymbol{q}}_n^{OO} = \left[\bar{\mathbf{R}}_1^O\right] \begin{bmatrix} \left[-\tilde{\mathbf{r}}_A^{11}\right] \\ \vdots \\ \left[-\tilde{\mathbf{r}}_N^{11}\right] \end{bmatrix} \left[\mathbf{R}_O^1\right] \ddot{\boldsymbol{q}}_1^{OO} + \left[\bar{\mathbf{R}}_1^O\right] \ddot{\boldsymbol{q}}_n^{11} + \left[\bar{\mathbf{R}}_1^O\right] \left[\dot{\mathbf{q}}_n^2\right] \tag{3.19}$$

where $\left[\bar{\mathbf{R}}_1^O\right]$ is a 6N by 6N matrix containing repetitions of the rotation matrix $\left[\mathbf{R}_1^O\right]$ on the diagonal and $\left[\dot{\mathbf{q}}_n^2\right]$ is a column vector built for all nodes in the FEM mesh:

$$\left[\dot{\mathbf{q}}_n^2\right] = \begin{bmatrix} \left[\dot{\mathbf{q}}_A^2\right] \\ \vdots \\ \left[\dot{\mathbf{q}}_N^2\right] \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{r}_A^{11} + 2\tilde{\boldsymbol{\omega}}_1^{1O} \dot{\boldsymbol{r}}_A^{11} \\ \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{\omega}_A^{11} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{r}_N^{11} + 2\tilde{\boldsymbol{\omega}}_1^{1O} \dot{\boldsymbol{r}}_N^{11} \\ \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{\omega}_N^{11} \end{bmatrix} \end{bmatrix} \tag{3.20}$$

The second matrix in (3.19) contains the rigid body modes. Because the rigid and flexible movement is decoupled by use of the floating frame definition, the rigid and flexible movement can be described using rigid body modes and flexible modes respectively [1, 2, 11]. The rigid body modes describe unit translations and rotations of the body without any deformation and the flexible modes describe deformations of the body [1]. Figure 3.3 shows the rigid body modes for a two-dimensional body. The rigid body modes for the mesh containing N nodes are expressed as:

$$\boldsymbol{\Phi}_{rig} = \begin{bmatrix} \left[-\tilde{\mathbf{r}}_A^{11}\right] \\ \vdots \\ \left[-\tilde{\mathbf{r}}_N^{11}\right] \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \mathbf{1} & -\tilde{\boldsymbol{r}}_A^{11} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \mathbf{1} & -\tilde{\boldsymbol{r}}_N^{11} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \end{bmatrix} \tag{3.21}$$

where $\boldsymbol{\Phi}_{rig}$ are the rigid body modes in which the identity matrices represent the unit displacements and rotations and the skew matrices denote the deflection of the nodes for the unit displacements. The skew matrix of point A is defined as:

$$-\tilde{\boldsymbol{r}}_A^{11} = \begin{bmatrix} 0 & z_A^{11} & -y_A^{11} \\ -z_A^{11} & 0 & x_A^{11} \\ y_A^{11} & -x_A^{11} & 0 \end{bmatrix} \tag{3.22}$$

where x, y and z are the deflections along the x, y and z axis respectively.

Figure 3.3: Two dimensional rigid body modes.

Substitution of the definition of the rigid body modes in the acceleration vector (3.19) yields:

$$\ddot{\boldsymbol{q}}_n^{OO} = \left[\bar{\mathbf{R}}_1^O\right] \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] \ddot{\boldsymbol{q}}_1^{OO} + \left[\bar{\mathbf{R}}_1^O\right] \ddot{\boldsymbol{q}}_n^{11} + \left[\bar{\mathbf{R}}_1^O\right] \left[\dot{\mathbf{q}}_n^2\right] \tag{3.23}$$

By combining the first two terms on the right hand side of (3.23) the formulation of the vector of accelerations of every node in the mesh:

$$\ddot{\boldsymbol{q}}_n^{OO} = \left[\left[\bar{\mathbf{R}}_1^O\right] \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] \quad \left[\bar{\mathbf{R}}_1^O\right]\right] \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{q}}_n^{11} \end{bmatrix} + \left[\bar{\mathbf{R}}_1^O\right] \left[\dot{\mathbf{q}}_n^2\right] \tag{3.24}$$

is obtained.

The vector of all accelerations (3.24) maybe be substituted in the non-linear equation of motion (3.7) to obtain:

$$\left[\left[\bar{\mathbf{R}}_1^O\right] \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] \quad \left[\bar{\mathbf{R}}_1^O\right] \boldsymbol{M}_{FE}\right] \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{q}}_n^{11} \end{bmatrix} + \left[\bar{\mathbf{R}}_1^O\right] \boldsymbol{M}_{FE} \left[\dot{\mathbf{q}}_n^2\right] + \left[\bar{\mathbf{R}}_1^O\right] \boldsymbol{K}_{FE} \boldsymbol{q}_n^{11} = \left[\bar{\mathbf{R}}_1^O\right] \boldsymbol{F}^1 \tag{3.25}$$

By pre-multiplication of (3.25) with $\begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \left[\bar{\mathbf{R}}_O^1\right] \\ \left[\bar{\mathbf{R}}_O^1\right] \end{bmatrix}$, the first matrix is transformed to the shape of the global mass matrix where the first term can be recognized as the submatrix of the rigid motion:

$$\begin{aligned} \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] & \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \\ \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] & \boldsymbol{M}_{FE} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{q}}_n^{11} \end{bmatrix} + \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \mathbf{1} \end{bmatrix} \boldsymbol{M}_{FE} \left[\dot{\mathbf{q}}_n^2\right] \\ + \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \mathbf{1} \end{bmatrix} \boldsymbol{K}_{FE} \boldsymbol{q}_n^{11} = \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \mathbf{1} \end{bmatrix} \boldsymbol{F}^1 \end{aligned} \tag{3.26}$$

It should be noted that the number of coordinates is six larger compared to the number of equations in (3.26) because $\boldsymbol{q}_n^{11}$ contains the rigid body movement also expressed by $\boldsymbol{q}_1^{OO}$. As stated, the flexible movement can be described using flexible modes by expressing the flexible motion of the body in the local frame as the product of a set of flexible mode shapes with an equal number of generalized flexible coordinates:

$$\boldsymbol{q}_n^{11} = \boldsymbol{\Phi}_{flex} \boldsymbol{\eta} \tag{3.27}$$

where $\boldsymbol{\Phi}_{flex}$ is a matrix containing the flexible modes and $\boldsymbol{\eta}$ is the vector of time dependent generalized flexible coordinates [1, 2, 11]. The set of flexible modes forms an orthogonal basis, which is important to note because an orthogonal basis can be truncated to achieve reduction of the model. By excluding the rigid body modes from the set of flexible modes (which will be discuss in section 3.2), the number of flexible coordinates is reduced such that the number of equations in (3.26) is equal to the number of coordinates. One can define the accelerations of the flexible motion in terms of flexible modes by finding the second derivative of (3.27) with respect to time:

$$\ddot{\boldsymbol{q}}_n^{11} = \boldsymbol{\Phi}_{flex} \ddot{\boldsymbol{\eta}} \tag{3.28}$$

where $\ddot{\boldsymbol{\eta}}$ are the time dependent generalized flexible accelerations. Substitution of the expression of the accelerations of the flexible movement (3.28) in the equation of motion (3.26) gives:

$$\begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] & \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \\ \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] & \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{\eta}} \end{bmatrix} + \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \mathbf{1} \end{bmatrix} \boldsymbol{M}_{FE} \left[\dot{\mathbf{q}}_n^2\right] + \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \mathbf{1} \end{bmatrix} \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \boldsymbol{\eta} = \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \mathbf{1} \end{bmatrix} \boldsymbol{F}^1 \tag{3.29}$$

The bottom row of equation (3.29) is projected by multiplication with $\boldsymbol{\Phi}_{flex}^T$:

$$\begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] & \left[\boldsymbol{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \\ \boldsymbol{\Phi}_{flex}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] & \boldsymbol{\Phi}_{flex}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{\eta}} \end{bmatrix} + \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{M}_{FE} \left[\dot{\mathbf{q}}_n^2\right] + \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \boldsymbol{\eta} = \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{F}^1 \tag{3.30}$$

such that the global mass matrix is obtained which has on the diagonal the submatrices of the rigid motion:

$$\boldsymbol{M}_{rr} = \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] \tag{3.31}$$

and the flexible motion:

$$\boldsymbol{M}_{ff} = \boldsymbol{\Phi}_{flex}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \tag{3.32}$$

respectively [1, 2]. The remaining two matrices are the coupling terms between the rigid and flexible movements:

$$\boldsymbol{M}_{rf} = \left[\boldsymbol{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \qquad \boldsymbol{M}_{fr} = \boldsymbol{\Phi}_{flex}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} \left[\mathbf{R}_O^1\right] \tag{3.33}$$

[1, 2]. Similar to the mass matrix, the third term of (3.30):

$$\begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \boldsymbol{\eta} \tag{3.34}$$

can be rewritten to obtain the global stiffness matrix. Knowing any projection of the stiffness matrix onto the rigid modes equals zero:

$$\boldsymbol{\Phi}_{rig}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{rig} = \mathbf{0} \qquad \boldsymbol{\Phi}_{rig}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} = \mathbf{0} \qquad \boldsymbol{\Phi}_{flex}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{rig} = \mathbf{0} \tag{3.35}$$

allows one to write (3.34) as:

$$\begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\Phi}_{rig}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{rig}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \\ \boldsymbol{\Phi}_{flex}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{flex}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_1^{OO} \\ \boldsymbol{\eta} \end{bmatrix} \tag{3.36}$$

which, similar to the global mass matrix, has on the diagonal the submatric of the rigid motion and the submatrix of the flexible motion and on the off-diagonal the coupling terms between the rigid and flexible motion. Substitution of the expression for the stiffness term (3.36) into (3.30) gives the equation of motion:

$$\begin{bmatrix} \left[\mathbf{R}_1^O\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Phi}_m^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_m \begin{bmatrix} \left[\mathbf{R}_O^1\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{\eta}} \end{bmatrix} + \begin{bmatrix} \left[\mathbf{R}_1^O\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Phi}_m^T \boldsymbol{M}_{FE} \left[\dot{\mathbf{q}}_n^2\right] + \boldsymbol{\Phi}_m^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_m \begin{bmatrix} \boldsymbol{q}_1^{OO} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \left[\mathbf{R}_1^O\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Phi}^T \boldsymbol{F}^1 \tag{3.37}$$

in which the rigid and flexible modes are combined in one matrix $\boldsymbol{\Phi}_m$:

$$\boldsymbol{\Phi}_m = \begin{bmatrix} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{flex} \end{bmatrix} \tag{3.38}$$

The quadratic velocity term $\boldsymbol{Q}_v$ can be recognized from (3.37) as:

$$\boldsymbol{Q}_v = \begin{bmatrix} \left[\mathbf{R}_1^O\right] \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{M}_{FE} \left[\dot{\mathbf{q}}_n^2\right] \tag{3.39}$$

Unless problems of rapid rotational movement are considered, the quadratic velocity terms can typically be neglected because the velocities and angular velocities are relatively small [1, 2]. Therefore, the quadratic terms will henceforth be discarded. In case one must include the quadratic velocity terms, Appendix A expands on (3.39) to show how the terms are evaluated.

By discarding the quadratic velocity terms in (3.37), the equation of motion can be written as:

$$\begin{bmatrix} \left[\mathbf{R}_1^O\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Phi}_m^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_m \begin{bmatrix} \left[\mathbf{R}_O^1\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{\eta}} \end{bmatrix} + \boldsymbol{\Phi}_m^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_m \begin{bmatrix} \boldsymbol{q}_1^{OO} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \left[\mathbf{R}_1^O\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Phi}_m^T \boldsymbol{F}^1 \tag{3.40}$$

For a more compact notation, one may define the global mass matrix, stiffness matrix, resultant force vector, generalized coordinates and accelerations as:

$$\boldsymbol{M} = \begin{bmatrix} \left[\mathbf{R}_1^O\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Phi}_m^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_m \begin{bmatrix} \left[\mathbf{R}_O^1\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3.41}$$

$$\boldsymbol{K} = \boldsymbol{\Phi}_m^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_m \tag{3.42}$$

$$\boldsymbol{Q} = \begin{bmatrix} \left[\mathbf{R}_1^O\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{\Phi}_m^T \boldsymbol{F}^1 \tag{3.43}$$

$$\boldsymbol{q} = \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{\eta}} \end{bmatrix} \tag{3.44}$$

$$\ddot{\boldsymbol{q}} = \begin{bmatrix} \ddot{\boldsymbol{q}}_1^{OO} \\ \ddot{\boldsymbol{\eta}} \end{bmatrix} \tag{3.45}$$

respectively. Therefore, (3.40) can be rewritten as:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} = \boldsymbol{Q} \tag{3.46}$$

In addition to the equation of motion, the constraint equations (3.2) are written using the floating frame formulation as well by expressing the constraints in terms of the generalized coordinates:

$$\boldsymbol{\Phi}(\boldsymbol{q}, t) = \mathbf{0} \tag{3.47}$$

The first derivative of the constraint equations (3.47) with respect to time is defined as:

$$\dot{\boldsymbol{\Phi}} = \boldsymbol{\Phi}_{\boldsymbol{q}} \dot{\boldsymbol{q}} + \boldsymbol{\Phi}_t = \mathbf{0} \tag{3.48}$$

where $\boldsymbol{\Phi}_{\boldsymbol{q}}$ is the Jacobian of the constraints and $\boldsymbol{\Phi}_t$ is the derivative of the constraints to time. By taking the derivative again, the acceleration equation is found:

$$\ddot{\boldsymbol{\Phi}} = \left[\boldsymbol{\Phi}_{\boldsymbol{q}} \dot{\boldsymbol{q}}\right]_{\boldsymbol{q}} \dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{\boldsymbol{q}t} \dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{\boldsymbol{q}} \ddot{\boldsymbol{q}} + \boldsymbol{\Phi}_{t\boldsymbol{q}} \dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{tt} = \mathbf{0} \tag{3.49}$$

where $\left[\boldsymbol{\Phi}_{\boldsymbol{q}} \dot{\boldsymbol{q}}\right]_{\boldsymbol{q}}$ is a matrix of partial derivatives of $\boldsymbol{\Phi}_{\boldsymbol{q}} \dot{\boldsymbol{q}}$ with respect to the generalized coordinates, $\boldsymbol{\Phi}_{\boldsymbol{q}t}$ is the derivative of the Jacobian with respect to time and $\boldsymbol{\Phi}_{tt}$ is the vector of second derivatives of the constraints with respect to time. By defining:

$$\boldsymbol{\gamma} = -\left[\boldsymbol{\Phi}_{\boldsymbol{q}} \dot{\boldsymbol{q}}\right]_{\boldsymbol{q}} \dot{\boldsymbol{q}} - 2\boldsymbol{\Phi}_{\boldsymbol{q}t} \dot{\boldsymbol{q}} - \boldsymbol{\Phi}_{tt} \tag{3.50}$$

the acceleration equation can be written in a more compact notation:

$$\boldsymbol{\Phi}_{\boldsymbol{q}} \ddot{\boldsymbol{q}} = \boldsymbol{\gamma} \tag{3.51}$$

The equation of motion (3.46) and the acceleration equation (3.51) can be combined to obtain the constrained equation of motion [1, 2, 11]. However, the remaining problem is that the force vector $\boldsymbol{Q}$ is only partly known, because it contains the known applied forces $\boldsymbol{Q}_A$ and unknown constraint forces $\boldsymbol{Q}_C$:

$$\boldsymbol{Q} = \boldsymbol{Q}_A + \boldsymbol{Q}_C \tag{3.52}$$

In order to efficiently solve the equations, the applied and constraint forces must be separated. One can first apply the principle of virtual work on the equation on motion (3.46):

$$\delta \boldsymbol{q}^T(\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} - \boldsymbol{Q}) = \boldsymbol{0} \qquad \forall \delta \boldsymbol{q} \tag{3.53}$$

where $\delta \boldsymbol{q}$ is the virtual displacement vector. The force vector can be separated according to (3.52):

$$\delta \boldsymbol{q}^T(\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} - \boldsymbol{Q}_A) - \delta \boldsymbol{q}^T\boldsymbol{Q}_C = \boldsymbol{0} \qquad \forall \delta \boldsymbol{q} \tag{3.54}$$

If the virtual displacement satisfies the constraints, the constraint forces do not perform any virtual work:

$$\delta \boldsymbol{q}^T\boldsymbol{Q}_C = \boldsymbol{0} \qquad \forall \delta \boldsymbol{q} \tag{3.55}$$

which is applied to (3.54) to find:

$$\delta \boldsymbol{q}^T(\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} - \boldsymbol{Q}_A) = \boldsymbol{0} \qquad \forall \delta \boldsymbol{q} \tag{3.56}$$

If the constraints must be satisfied, the variation of the constraints must be as well:

$$\delta \boldsymbol{\Phi} = \boldsymbol{\Phi}_{\boldsymbol{q}} \delta \boldsymbol{q} = \boldsymbol{0} \qquad \forall \delta \boldsymbol{q} \tag{3.57}$$

As a result, two equations ((3.56) and (3.57)) are found which must be satisfied and can be combined using the Lagrange multiplier theorem [1, 2, 11]. Applying the theorem on the transposed of (3.56) and (3.57) yields:

$$(\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} - \boldsymbol{Q}_A)^T \delta \boldsymbol{q} + \boldsymbol{\lambda}^T \boldsymbol{\Phi}_{\boldsymbol{q}} \delta \boldsymbol{q} = \boldsymbol{0} \qquad \forall \delta \boldsymbol{q} \tag{3.58}$$

Dividing (3.58) by the virtual displacement vector results in an expression of the equation of motion in which the known applied forces and unknown constraint forces are separated:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} + {\boldsymbol{\Phi}_{\boldsymbol{q}}}^T \boldsymbol{\lambda} = \boldsymbol{Q}_A \tag{3.59}$$

which shows the relation between the Lagrange multipliers and the constraint forces:

$$\boldsymbol{Q}_C = -{\boldsymbol{\Phi}_{\boldsymbol{q}}}^T \boldsymbol{\lambda} \tag{3.60}$$

The equation of motion (3.59) and the acceleration equation (3.51), which are both given in terms of the generalized coordinates, can be combined to form the constrained equation of motion:

$$\begin{bmatrix} \boldsymbol{M} & {\boldsymbol{\Phi}_{\boldsymbol{q}}}^T \\ \boldsymbol{\Phi}_{\boldsymbol{q}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q} \\ \boldsymbol{\gamma} \end{bmatrix} \tag{3.61}$$

In summary, (3.61) is the constrained equation of motion with the following variables:

- $\boldsymbol{q}$ are the generalized coordinates of each body
- $\boldsymbol{M}$ is the mass matrix: $\boldsymbol{M} = \boldsymbol{\Phi}_m^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_m = \begin{bmatrix} \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{rig}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \\ \boldsymbol{\Phi}_{flex}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{flex}^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_{flex} \end{bmatrix}$
- $\boldsymbol{\Phi}_m = \begin{bmatrix} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{flex} \end{bmatrix}$ are the rigid and flexible mode shapes.
- $\boldsymbol{\Phi}_{\boldsymbol{q}}$ are the derivatives of the constraint equations ($\boldsymbol{\Phi}(\boldsymbol{q}) = \boldsymbol{0}$) with respect to $\boldsymbol{q}$, also called the Jacobian
- $\boldsymbol{Q}_A$ are the applied forces: $\boldsymbol{Q}_A = \boldsymbol{\Phi}_m^T \boldsymbol{Q}_{A,node} = \begin{bmatrix} \boldsymbol{\Phi}_{rig}^T \boldsymbol{Q}_{A,node} \\ \boldsymbol{\Phi}_{flex}^T \boldsymbol{Q}_{A,node} \end{bmatrix}$
- $\boldsymbol{K}$ is the stiffness matrix: $\boldsymbol{K} = \boldsymbol{\Phi}_m^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_m = \begin{bmatrix} \boldsymbol{\Phi}_{rig}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{rig}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \\ \boldsymbol{\Phi}_{flex}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{rig} & \boldsymbol{\Phi}_{flex}^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_{flex} \end{bmatrix}$
- $-{\boldsymbol{\Phi}_{\boldsymbol{q}}}^T \boldsymbol{\lambda}$ are the constraint forces
- $\boldsymbol{\gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the right hand side of the acceleration equation defined as $\boldsymbol{\gamma} \equiv -[\boldsymbol{\Phi}_{\boldsymbol{q}} \dot{\boldsymbol{q}}]_{\boldsymbol{q}} \dot{\boldsymbol{q}} - 2\boldsymbol{\Phi}_{\boldsymbol{q}_t} \dot{\boldsymbol{q}} - \boldsymbol{\Phi}_{tt}$

3.2 Discretization in space

The flexible motion is discretized in space using a set of orthonormal flexible modes as given in (3.27). The finite element interpolation functions are used as mode shapes [1]. Each mode describes one node being moved one unit in one direction, while the remaining nodes remain stationary. The elements directly attached to the displaced node are shaped according to the interpolation functions. Figure 3.4 shows the flexible modes in which the nodes are displaced in y-direction.

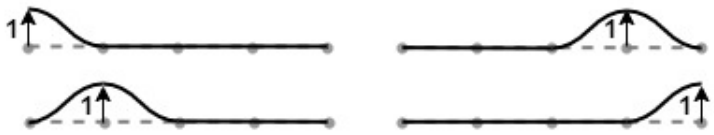


Figure 3.4: Unreduced flexible modes corresponding to a unit displacement in y-direction.

As stated, the rigid body modes must be removed from the set of flexible mode shapes, because the rigid body modes are already included separately in the constrained equation of motion. Therefore, allowing the flexible modes to form rigid body modes leads to singularities [1, 2]. The most common way to eliminate the rigid body modes is to fix the displacement of the node in the center of mass, such that the combination of all flexible modes with a unit displacement in one direction no longer forms a rigid body motion [1, 2]. Therefor, Figure 3.4 does not contain a mode corresponding to displacement of the node in the center of mass.

The disadvantage of using the FE interpolation functions as flexible modes is the large number of modes involved. For every node, except the one in the floating frame, six flexible modes are defined in a three dimensional environment. Adding the six rigid modes results in a set of modes equal to the number of nodes times the number of dimensions. Because a flexible coordinate is assigned to all modes, the set of generalized coordinates is large and solving the constrained equation of motion becomes time consuming [1]. However, because the modes shapes form an orthogonal basis, it is possible to truncate the set of flexible modes to obtain a reduced model. Different possibilities to achieve reduction in the discretization in space will be addressed in the next chapter.

3.3 Discretization in time

To perform a flexible multibody dynamic simulation over time, the constrained equation of motion must be integrated. For convenience, the system of equations is repeated:

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{\Phi_q}^T \\ \boldsymbol{\Phi_q} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q} \\ \boldsymbol{\gamma} \end{bmatrix} \tag{3.62}$$

To perform numerical integration, an integration scheme is required. Often the integration scheme is combined with Newton-Raphson, which is a method used to ensure the solutions obtained by the integration satisfy that the constraints [1, 11].

3.3.1 Time integration schemes.

Before applying time integration on the constrained equation of motion, one must note that the system is not an ODE but a DAE (differential-algebraic equations) and is of second order. The DAE can be integrated directly using forward Euler [1, 11], where the accelerations at one time step can be used to obtain the velocities at the next time step:

$$\dot{\boldsymbol{q}}(t_{i+1}) = \dot{\boldsymbol{q}}(t_i) + \ddot{\boldsymbol{q}}(t_{i+1})\Delta t \tag{3.63}$$

where t_i and t_{i+1} represent two consecutive time steps and Δt the size of the time step. Subsequently, explicit Euler can be applied to find the positions:

$$\boldsymbol{q}(t_{i+1}) = \boldsymbol{q}(t_i) + \dot{\boldsymbol{q}}(t_{i+1})\Delta t \tag{3.64}$$

The same combination of forward and explicit Euler can be used to integrate the Lagrange multipliers.

Alternatively, the DAE can be rewritten to a first order ODE [12] and solved using explicit Runge-Kutta. The constrained equation of motion (3.61) can be written as a first order set of equations:

$$\begin{cases} \dot{\boldsymbol{q}} &= \boldsymbol{v} & \text{(3.65a)} \\ \boldsymbol{M}\dot{\boldsymbol{v}} &= \boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q} - \boldsymbol{\Phi_q}^T\boldsymbol{\lambda} & \text{(3.65b)} \\ \boldsymbol{0} &= \boldsymbol{\Phi_q}\dot{\boldsymbol{v}} - \boldsymbol{\gamma} & \text{(3.65c)} \end{cases}$$

Multiplication of (3.65b) with the inverse of the mass matrix yields an expression for $\dot{\boldsymbol{v}}$:

$$\dot{\boldsymbol{v}} = \boldsymbol{M}^{-1}(\boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q} - \boldsymbol{\Phi_q}^T\boldsymbol{\lambda}) \tag{3.66}$$

One can substitute the definition for $\dot{\boldsymbol{v}}$ in (3.65c) to find:

$$\boldsymbol{0} = \boldsymbol{\Phi_q}\boldsymbol{M}^{-1}(\boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q} - \boldsymbol{\Phi_q}^T\boldsymbol{\lambda}) - \boldsymbol{\gamma} \tag{3.67}$$

which can be manipulated to obtain an expression for the Lagrange multipliers from which the acceleration terms have been removed:

$$\boldsymbol{\lambda} = \left(\boldsymbol{\Phi_q}\boldsymbol{M}^{-1}\boldsymbol{\Phi_q}^T\right)^{-1}\left(\boldsymbol{\Phi_q}\boldsymbol{M}^{-1}(\boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q}) - \boldsymbol{\gamma}\right) \tag{3.68}$$

Substitution of the expression for $\boldsymbol{\lambda}$ in (3.65a) and (3.65b) results in a set of first order ODE's for $\boldsymbol{q}$ and $\boldsymbol{v}$ also called the state space form:

$$\begin{cases} \dot{\boldsymbol{q}} = \boldsymbol{v} & \text{(3.69a)} \\ \dot{\boldsymbol{v}} = \boldsymbol{M}^{-1}(\boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q}) - \boldsymbol{M}^{-1}\boldsymbol{\Phi_q}^T\left(\boldsymbol{\Phi_q}\boldsymbol{M}^{-1}\boldsymbol{\Phi_q}^T\right)^{-1}\left(\boldsymbol{\Phi_q}\boldsymbol{M}^{-1}(\boldsymbol{Q}_A - \boldsymbol{K}\boldsymbol{q}) - \boldsymbol{\gamma}\right) & \text{(3.69b)} \end{cases}$$

which can be solved using Runge-Kutta. It should be noted that the state space form and the original DAE are not equivalent. Drift problems can occur when solving the state space form which might result in solutions which violate the constraints for the model [12]. However, by combining the numerical integration with Newton-Raphson, potential problems will be recognised.

The state space form (3.69a) and (3.69b) is a set of ODE's for every generalized coordinate. Alternatively, one can classify the generalized coordinates of the system as independent or dependent coordinates [1, 13]. The number of independent coordinates is equal to the number of degrees of freedom, while the remaining dependent coordinates are related to the independent coordinates by the constraints imposed on the system. Through the distinction between the coordinates, a set of first order ODE's of the independent coordinates only can be obtained called the embedded formulation [13]. To find the embedded formulation, the constrained equations of motion must be written in terms of independent generalized coordinates $\boldsymbol{q}_i$ and dependent generalized coordinates $\boldsymbol{q}_d$:

$$\begin{bmatrix} \boldsymbol{M}_{dd} & \boldsymbol{M}_{di} & \boldsymbol{\Phi}_{\boldsymbol{q}_d}^T \\ \boldsymbol{M}_{id} & \boldsymbol{M}_{ii} & \boldsymbol{\Phi}_{\boldsymbol{q}_i}^T \\ \boldsymbol{\Phi}_{\boldsymbol{q}_d} & \boldsymbol{\Phi}_{\boldsymbol{q}_i} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_d \\ \ddot{\boldsymbol{q}}_i \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_d - \boldsymbol{K}_{dd}\boldsymbol{q}_d - \boldsymbol{K}_{di}\boldsymbol{q}_i \\ \boldsymbol{Q}_i - \boldsymbol{K}_{id}\boldsymbol{q}_d - \boldsymbol{K}_{ii}\boldsymbol{q}_i \\ \boldsymbol{\gamma} \end{bmatrix} \tag{3.70}$$

where $\boldsymbol{M}_{dd}$, $\boldsymbol{M}_{di}$, $\boldsymbol{M}_{id}$ and $\boldsymbol{M}_{ii}$ are partitions of the mass matrix, $\boldsymbol{K}_{dd}$, $\boldsymbol{K}_{di}$, $\boldsymbol{K}_{id}$ and $\boldsymbol{K}_{ii}$ are partitions of the stiffness matrix, $\boldsymbol{Q}_d$ and $\boldsymbol{Q}_i$ are partitions of the applied force vector $\boldsymbol{Q}_A$ and $\boldsymbol{\Phi}_{\boldsymbol{q}_d}$ and $\boldsymbol{\Phi}_{\boldsymbol{q}_i}$ are the derivatives of the constraint equations with respect to the dependent and independent generalized coordinates respectively [13]. It should be noted that $\boldsymbol{M}_{dd}$, $\boldsymbol{M}_{ii}$ and $\boldsymbol{Q}_d$ are square matrices which can therefore generally be inverted. The first row of (3.70) can thus be used to find an expression for $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda} = (\boldsymbol{\Phi}_{\boldsymbol{q}_d}^T)^{-1}(\boldsymbol{Q}_d - \boldsymbol{K}_{dd}\boldsymbol{q}_d - \boldsymbol{K}_{di}\boldsymbol{q}_i - \boldsymbol{M}_{dd}\ddot{\boldsymbol{q}}_d - \boldsymbol{M}_{di}\ddot{\boldsymbol{q}}_i) \tag{3.71}$$

One can use the third row of (3.70) to find the relation between the independent and dependent accelerations:

$$\ddot{\boldsymbol{q}}_d = (\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}(\boldsymbol{\gamma} - \boldsymbol{\Phi}_{\boldsymbol{q}_i}\ddot{\boldsymbol{q}}_i) \tag{3.72}$$

Substitution of (3.71) and (3.72) in the second row of (3.70) yields the second order ODE's for the independent generalized coordinates:

$$\begin{aligned}&\Big(-\boldsymbol{M}_{id}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\Phi}_{\boldsymbol{q}_i} + \boldsymbol{M}_{ii} - \boldsymbol{\Phi}_{\boldsymbol{q}_i}^T(\boldsymbol{\Phi}_{\boldsymbol{q}_d}^T)^{-1}(\boldsymbol{M}_{di} - \boldsymbol{M}_{dd}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\Phi}_{\boldsymbol{q}_i})\Big)\ddot{\boldsymbol{q}}_i = \\ &-\boldsymbol{M}_{id}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\gamma} - \boldsymbol{\Phi}_{\boldsymbol{q}_i}^T(\boldsymbol{\Phi}_{\boldsymbol{q}_d}^T)^{-1}(\boldsymbol{Q}_d - \boldsymbol{K}_{dd}\boldsymbol{q}_d - \boldsymbol{K}_{di}\boldsymbol{q}_i - \boldsymbol{M}_{dd}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\gamma}) + \boldsymbol{Q}_i - \boldsymbol{K}_{id}\boldsymbol{q}_d - \boldsymbol{K}_{ii}\boldsymbol{q}_i\end{aligned} \tag{3.73}$$

For a more compact notation, one can define the left hand and right hand sides of (3.73) as

$$\hat{\boldsymbol{M}} = -\boldsymbol{M}_{id}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\Phi}_{\boldsymbol{q}_i} + \boldsymbol{M}_{ii} - \boldsymbol{\Phi}_{\boldsymbol{q}_i}^T(\boldsymbol{\Phi}_{\boldsymbol{q}_d}^T)^{-1}(\boldsymbol{M}_{di} - \boldsymbol{M}_{dd}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\Phi}_{\boldsymbol{q}_i}) \tag{3.74}$$

and

$$\hat{\boldsymbol{Q}} = -\boldsymbol{M}_{id}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\gamma} - \boldsymbol{\Phi}_{\boldsymbol{q}_i}^T(\boldsymbol{\Phi}_{\boldsymbol{q}_d}^T)^{-1}(\boldsymbol{Q}_d - \boldsymbol{K}_{dd}\boldsymbol{q}_d - \boldsymbol{K}_{di}\boldsymbol{q}_i - \boldsymbol{M}_{dd}(\boldsymbol{\Phi}_{\boldsymbol{q}_d})^{-1}\boldsymbol{\gamma}) + \boldsymbol{Q}_i - \boldsymbol{K}_{id}\boldsymbol{q}_d - \boldsymbol{K}_{ii}\boldsymbol{q}_i \tag{3.75}$$

respectively. Therefore, (3.73) can be rewritten as:

$$\hat{\boldsymbol{M}}\ddot{\boldsymbol{q}}_i = \hat{\boldsymbol{Q}} \tag{3.76}$$

which is a set of second order ODE's of the independent generalized coordinates and are written in state space form as:

$$\begin{cases} \dot{\boldsymbol{q}}_i = \boldsymbol{v}_i & \text{(3.77a)} \\ \dot{\boldsymbol{v}}_i = \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{Q}} & \text{(3.77b)} \end{cases}$$

The system (3.77a) and (3.77b) is the embedded formation of the constrained equations of motion which can be numerically integrated using Runge-Kutta to obtain the independent generelized coordinates. The dependent coordinates are found by (3.72) and application of Newton-Raphson [13]. The advantage of the embedded formulation over the state space form of all generalized coordinates (3.69a) and (3.69b) is that the matrix which must be inverted in the embedded formulation ($\hat{\boldsymbol{M}}$) is relatively small compared to the matrix $\boldsymbol{M}$ in (3.69b) [13]. Because both $\hat{\boldsymbol{M}}$ and $\boldsymbol{M}$ must be calculated and inverted at every time step, the embedded formulation is expected to be more efficient.

Runge-Kutta can be implemented as a fourth or fifth order scheme using available toolboxes in programming software, while forward Euler has a first order accuracy [14]. Furthermore, the toolboxes allow for adaptive applications of Runge-Kutta with an adaptive time step [14]. Therefore, applying Runge-Kutta on either the state space form for all generalized coordinates or the embedded formulation is expected to result in a more exact solution compared to application of forward and explicit Euler on the DAE.

3.3.2 Newton-Raphson

The solution for the position vector $\boldsymbol{q}$ found by numerical integration likely does not satisfy the constraint equations, because numerical integration is not an exact method [1, 2, 11]. To assure the constraint equations are satisfied, the Newton-Raphson method is applied on the dependent generalized coordinates such that the dependent generalized coordinates are updated to satisfy the constraint equations while the independent generalized coordinates retain the values obtained through numerical integration [1, 2, 11]. The exact solution $\boldsymbol{q}^{ex}$ at some time instance satisfies the constraints such that no error is left:

$$\boldsymbol{\Phi}(\boldsymbol{q}^{ex}) = \boldsymbol{0} \tag{3.78}$$

However, a solution of the numerical integration $\boldsymbol{q}^0$ likely does not and has an equation error $\boldsymbol{\epsilon}_e$:

$$\boldsymbol{\epsilon}_e = |\boldsymbol{\Phi}(\boldsymbol{q}^0)| \tag{3.79}$$

An improved approximation of the solution $\boldsymbol{q}^1$ is found using the first order Taylor series around $\boldsymbol{q}^0$:

$$\boldsymbol{\Phi}(\boldsymbol{q})|_{\boldsymbol{q}\approx\boldsymbol{q}^0} = \boldsymbol{\Phi}(\boldsymbol{q}^0) + (\boldsymbol{q} - \boldsymbol{q}^0)\boldsymbol{\Phi}_{\boldsymbol{q}}(\boldsymbol{q}^0) \tag{3.80}$$

[11]. Knowing the series should equal zero, one can rewrite (3.80) to obtain the expression for the improved approximation $\boldsymbol{q}^1$:

$$\boldsymbol{q}^1 = \boldsymbol{q}^0 - [\boldsymbol{\Phi}_{\boldsymbol{q}}(\boldsymbol{q}^0)]^{-1}\boldsymbol{\Phi}(\boldsymbol{q}^0) \tag{3.81}$$

Equation (3.81) must be iterated until the equation error is below a tolerance. Apart from the equation error, it is useful to consider the solution error $\boldsymbol{\epsilon}_s$ after every iteration as well. The solution error is expressed as:

$$\boldsymbol{\epsilon}_s = |\boldsymbol{q}^1 - \boldsymbol{q}^0| \tag{3.82}$$

The solution of the numerical integration should yield a result which is close to the exact solution, meaning the solution error should be small for the initial solution. If, after several Newton-Raphson iterations, the equation error is sufficiently small but the solution error is not, the improved solution has diverged away from the initial solution. Because the initial solution must be close to the exact solution, the improved solution should not be used if the solution error is large. Furthermore, it is possible for Newton-Raphson to not have solution convergence, which can be caught by implementing Newton-Raphson with a limited number of iterations.

In summary, the Newton-Raphson method should be implemented according to Algorithm 1.

Algorithm 1: Newton-Raphson method.

```
Initialization: i, q^i, ε_s;
while ε_e(q^i) > tolerance && ε_s(q^i) > tolerance && i < MaxSteps do
    q^{i+1} = q^i − [Φ_q(q^i)]^{-1} Φ(q^i) ;
    ε_s = |q^{i+1} − q^i| ;
    i = i + 1 ;
end
```

3.3.3 Updating the rotation matrix

In three dimensional simulations the rotation matrices are important, because the rotations about the three axes cannot be expressed as proper vectors. The order in which rotations are applied matters for the resulting configuration. The rotation matrices of the initial configuration can be determined. However, the rotation matrices must be updated at every time step to match the new configuration. So, at every step during integration and at every iteration of Newton-Raphson, the rotation matrix must be updated while the properties of a rotation matrix must be retained. The update can be performed by defining the change in angle over a time step Δt:

$$\Delta\boldsymbol{\theta} = \boldsymbol{\omega}(t_{i+1})\Delta t \tag{3.83}$$

where $\Delta\boldsymbol{\theta}$ is the change of the angles, $\boldsymbol{\omega}$ is the vector of angular velocities, t_{i+1} is time instance i+1 and Δt is the time step. One can update the rotation matrix using the change in angles:

$$\boldsymbol{R}_{t_{i+1}} = \Big(\Delta\boldsymbol{R}_x(\Delta\theta_x)\Delta\boldsymbol{R}_y(\Delta\theta_y)\Delta\boldsymbol{R}_z(\Delta\theta_z)\Big)\boldsymbol{R}_{t_i} \tag{3.84}$$

where $\boldsymbol{R}_{t_{i+1}}$ is the updated rotation matrix at time instance i+1, $\boldsymbol{R}_{t_i}$ is the previous rotation matrix at time instance i and $\Delta\theta_x$, $\Delta\theta_y$ and $\Delta\theta_z$ are the change in angle about the x, y and z axis respectively. $\Delta\boldsymbol{R}_x$, $\Delta\boldsymbol{R}_y$ and $\Delta\boldsymbol{R}_z$ are the rotation matrices about the x, y and z axis expressed as:

$$\Delta\boldsymbol{R}_x(\Delta\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\theta_x) & -\sin(\Delta\theta_x) \\ 0 & \sin(\Delta\theta_x) & \cos(\Delta\theta_x) \end{bmatrix} \tag{3.85}$$

$$\Delta\boldsymbol{R}_y(\Delta\theta_y) = \begin{bmatrix} \cos(\Delta\theta_y) & 0 & \sin(\Delta\theta_y) \\ 0 & 1 & 0 \\ -\sin(\Delta\theta_y) & 0 & \cos(\Delta\theta_y) \end{bmatrix} \tag{3.86}$$

$$\Delta \boldsymbol{R}_z(\Delta\theta_z) = \begin{bmatrix} \cos(\Delta\theta_z) & -\sin(\Delta\theta_z) & 0 \\ \sin(\Delta\theta_z) & \cos(\Delta\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.87}$$

respectively.

4 Model reduction

The flexible motion is discretized in space using the finite element interpolation functions as flexible modes shapes. The number of mode shape, and thus the number of generalized flexible coordinates, is therefore equal to the number of nodal coordinates in the mesh times the number of dimensions, excluding the node is the center of mass. The large number of coordinates makes the constrained equation of motion a high dimensional problem, causing a long evaluation time. Therefore, one requires model reduction, which can be achieved by physics-based methods or by data driven techniques. The use of finite interpolation functions as flexible modes will henceforth be referenced as the unreduced modes.

4.1 Free boundary modes

The use of free boundary modes is a physics-bases approach to achieve reduction of the number of generalized coordinates by using the natural vibration modes to discretize the flexible movement [1, 2]. The natural vibration modes of the system are defined as the eigenmodes in case no constraints are imposed on the body. The free boundary modes can be applied because the flexible motion is harmonic and can thus be represented by the natural vibration modes of the system. The modes are obtained by solving the eigenvalue problem:

$$(\boldsymbol{K}_{FE} - \lambda_i^2 \boldsymbol{M}_{FE})\boldsymbol{\Phi}_i = \boldsymbol{0} \tag{4.1}$$

where $\boldsymbol{K}_{FE}$ is the finite element stiffness matrix, $\boldsymbol{M}_{FE}$ is the finite element mass matrix and $\boldsymbol{\Phi}_i$ is the natural vibration mode corresponding to the natural frequency λ_i. The first eigenvalues obtained when solving (4.1) equal zero and correspond to rigid body modes. All modes corresponding to an eigenvalue of zero must be removed from the set of flexible modes to prevent singularity problems because the rigid body modes are included in the constrained equation of motion separately [1, 2]. Figure 4.1 shows three free boundary modes.

Figure 4.1: Three free boundary modes.

The set of natural vibration modes found using (4.1) is large. If all natural modes are included in the set of flexible modes, the dimension of the constrained equation of motion is therefore still high. However, the flexible behaviour typically only needs to be included in a limited sense [1], implicating that not all eigenmodes need to be included in the set of flexible modes. Higher order natural modes typically contribute deformations of smaller magnitude compared to lower order modes, since higher order modes require more energy for the deformation to be significant compared to lower order modes. By neglecting the higher order modes only the detailed flexible behaviour is not captured, while the dominant behaviour is included by the lower order mode shapes. Using solely the lower order natural vibration modes as flexible modes shapes achieves reduction of the dimensions of the constrained equation of motion.

The additional advantage of using free boundary modes is that the stiffness matrix and mass matrix used in the constrained equation of motion

$$\boldsymbol{K} = \boldsymbol{\Phi}_m^T \boldsymbol{K}_{FE} \boldsymbol{\Phi}_m \tag{4.2}$$

and

$$\boldsymbol{M} = \boldsymbol{\Phi}_m^T \boldsymbol{M}_{FE} \boldsymbol{\Phi}_m \tag{4.3}$$

are diagonal [1], further reducing the computation time.

4.2 Craig Bampton

Alternatively to free boundary modes, the Craig Bampton method can be applied. Craig Bampton is a physics-based method to reduce the number of flexible modes and thus the number of generalized flexible coordinates [1, 2]. One can classify the nodes in the mesh as either internal (i) or boundary (b) nodes. The boundary nodes are located at the interface point between different bodies or the fixed world as schematically

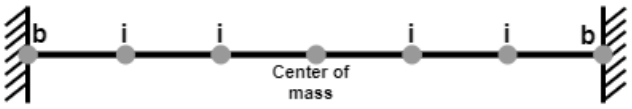


Figure 4.2: Schematic illustration of the boundary (b) and internal (i) nodes.

represented in Figure 4.2. Nodes which are not located at interface points, but to which forces are applied should be classified as boundary nodes as well.

The mode shapes are defined by the displacement of one boundary node one unit in one direction, while all other boundary nodes and the node in the center of mass remain fixed. Fixing the node in the center of mass is the most common way to eliminate rigid body modes from the set of flexible mode shapes, preventing singularities since the rigid body modes are included separately in the constrained equations of motion [1, 2]. Figure 4.3 shows the six Craig Bampton boundary modes of a bar of which the left and right most nodes are the boundary nodes and is considered in two dimensions.

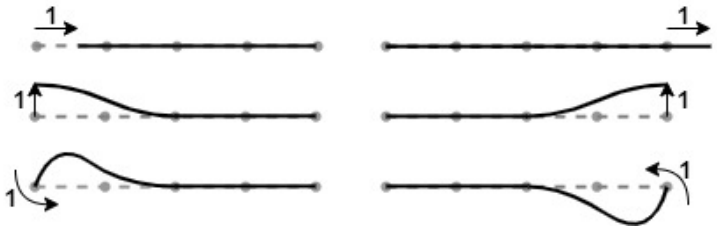


Figure 4.3: The boundary modes shapes by Craig Bampton of a two-dimensional bar of which the left and right most nodes are the boundary nodes.

The deformation shapes can be determined by solving the static FEM problem:

$$\boldsymbol{K}_{FE}\boldsymbol{q} = \boldsymbol{f} \tag{4.4}$$

where $\boldsymbol{K}_{FE}$ is the finite element stiffness matrix, $\boldsymbol{q}$ is the vector of generalized coordinates and $\boldsymbol{f}$ is a vector of applied forces. First, the stiffness matrix must be reordered into internal, boundary and coupling terms:

$$\begin{bmatrix} \boldsymbol{K}_{bb} & \boldsymbol{K}_{bi} \\ \boldsymbol{K}_{ib} & \boldsymbol{K}_{ii} \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_b \\ \boldsymbol{q}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_b \\ \boldsymbol{0} \end{bmatrix} \tag{4.5}$$

where $\boldsymbol{K}_{bb}$ is the section of the stiffness matrix representing the boundary nodes, $\boldsymbol{K}_{ii}$ is the section of the stiffness matrix representing the internal nodes, $\boldsymbol{K}_{bi}$ and $\boldsymbol{K}_{ib}$ are the sections of the stiffness matrix representing the coupling terms, $\boldsymbol{q}_b$ is the vector of the boundary coordinates, $\boldsymbol{q}_i$ is the vector of the internal coordinates and $\boldsymbol{Q}_b$ is the vector of applied forces. Using the second row of (4.5), the displacement of the internal nodes can be expressed in terms of the displacement of the boundary nodes:

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_b \\ \boldsymbol{q}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{1} \\ -\boldsymbol{K}_{ii}^{-1}\boldsymbol{K}_{ib} \end{bmatrix} \boldsymbol{q}_b \tag{4.6}$$

The matrix on the right hand side of (4.6) contains the mode shapes according to Craig Bampton:

$$\boldsymbol{\Phi}_{CB} = \begin{bmatrix} \boldsymbol{1} \\ -\boldsymbol{K}_{ii}^{-1}\boldsymbol{K}_{ib} \end{bmatrix} \tag{4.7}$$

where $\boldsymbol{\Phi}_{CB}$ denotes the Craig Bampton boundary modes. The identity matrix in the definition of the modes represents the unit displacements of the boundary nodes in each direction, while $-\boldsymbol{K}_{ii}^{-1}\boldsymbol{K}_{ib}$ represents the response of the internal nodes. Before the modes can be applied, the modes must be reordered such that the nodes are in the original configuration again and the displacement of the middle node should be added to the modes as zeros.

If a more detailed description of the flexible behaviour of the internal nodes is required, constrained eigenmodes might be added to the set of boundary modes [1]. The internal modes are found by solving the eigenvalue problem of the internal nodes while keeping all boundary nodes fixed:

$$(\boldsymbol{K}_{ii} - \lambda_i^2 \boldsymbol{M}_{ii})\boldsymbol{\Phi}_i = \boldsymbol{0} \tag{4.8}$$

where $\boldsymbol{\Phi}_i$ is the internal mode corresponding to the eigenfrequency λ_i. Figure 4.4 shows two internal modes. By adding the internal modes to the set of flexible modes shapes, the dimension of the constrained equation of motion is increased. Therefore, the internal modes should be included only if the detail is required.

Figure 4.4: Two of the internal mode shapes according to Craig Bampton.

So, using Craig Bampton results in a significant reduction of the dimensions compared to using the unreduced modes. The additional advantage of using Craig Bampton is that the Jacobian of the constraints is sparse [1], because the constraints are solely applied on the boundary nodes which have a unit displacement in only some of the mode shapes. Using internal modes does not complicate the Jacobian since the internal modes do not have a displacement at the boundary nodes. The fact that the Jacobian is sparse reduces the computation time further. For the reason that boundary nodes are involved in the floating frame definition, Craig Bampton is an attractive choice for the discretization of the flexible behaviour and therefore preferred over the free boundary modes in flexible multibody problem defined using floating frames [2].

4.3 Proper orthogonal decomposition

Instead of using a physics-based method to reduce the number of generalized flexible coordinates, the complete set of generalized coordinates in the constrained equations of motion formulated using unreduced modes can be reduced using proper orthogonal decomposition (POD) [4, 5]. POD is a data-driven reduction method which can be used to express the generalized coordinates as:

$$\boldsymbol{q}(t) = \boldsymbol{U}\boldsymbol{a}(t) + \bar{\boldsymbol{q}} \tag{4.9}$$

where $\boldsymbol{q}(t)$ is the vector of generalized coordinates, $\boldsymbol{U}$ is the orthogonal POD basis, $\boldsymbol{a}(t)$ is the vector of variables and $\bar{\boldsymbol{q}}$ is a vector of average values of the generalized coordinates [4, 5, 15, 16]. One can substitute the POD discretization (4.9) into the constrained equation of motion to obtain a system which can be solved for $\boldsymbol{a}$ instead of $\boldsymbol{q}$. For clarity, the constrained equation of motion is repeated here:

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{\Phi_q}^T(\boldsymbol{q}) \\ \boldsymbol{\Phi_q}(\boldsymbol{q}) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q} - \boldsymbol{K}\boldsymbol{q} \\ \boldsymbol{\gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} \tag{4.10}$$

4.3.1 Application of POD on FMBD: system level

POD can be applied on the constrained equations of motion on system level, meaning all generalized coordinates are approximated using one basis [4, 5]. To ensure the reduced system can be solved efficiently, the shape of the constrained equation of motion should be maintained: one vector containing all unknowns. The first step in rewriting the constrained equations of motion is to determine the derivatives of the projection of the generalized coordinates (4.9) with respect to time. The first derivative is defined as:

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{U}\dot{\boldsymbol{a}}(t) \tag{4.11}$$

and the second derivative is defined as:

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{U}\ddot{\boldsymbol{a}}(t) \tag{4.12}$$

One can substitute the projection and its derivatives in the constrained equation of motion to obtain:

$$\begin{bmatrix} \boldsymbol{M}\boldsymbol{U} & \boldsymbol{\Phi_q}^T(\boldsymbol{U}\boldsymbol{a} + \bar{\boldsymbol{q}}) \\ \boldsymbol{\Phi_q}(\boldsymbol{U}\boldsymbol{a} + \bar{\boldsymbol{q}})\boldsymbol{U} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{a}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q} - \boldsymbol{K}(\boldsymbol{U}\boldsymbol{a} + \bar{\boldsymbol{q}}) \\ \boldsymbol{\gamma}(\boldsymbol{U}\boldsymbol{a} + \bar{\boldsymbol{q}}), \boldsymbol{U}\dot{\boldsymbol{a}}) \end{bmatrix} \tag{4.13}$$

The top row of (4.13) is projected onto the transposed basis to find the symmetric expression:

$$\begin{bmatrix} \boldsymbol{U}^T\boldsymbol{M}\boldsymbol{U} & \boldsymbol{U}^T{\boldsymbol{\Phi}_{\boldsymbol{q}}}^T(\boldsymbol{U}\boldsymbol{a}+\bar{\boldsymbol{q}}) \\ \boldsymbol{\Phi}_{\boldsymbol{q}}(\boldsymbol{U}\boldsymbol{a}+\bar{\boldsymbol{q}})\boldsymbol{U} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{a}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}^T\boldsymbol{Q} - \boldsymbol{U}^T\boldsymbol{K}(\boldsymbol{U}\boldsymbol{a}+\bar{\boldsymbol{q}}) \\ \boldsymbol{\gamma}(\boldsymbol{U}\boldsymbol{a}+\bar{\boldsymbol{q}}, \boldsymbol{U}\dot{\boldsymbol{a}}) \end{bmatrix} \tag{4.14}$$

The result is a set of equations in the same shape as the original constrained equations of motion, in which the first vector contains the unknown accelerations and the Lagrange multipliers. Therefore, system (4.14) can be evaluated over time as described in Chapter 3.3. The resulting variables $\boldsymbol{a}$ and its derivatives can be projected back to $\boldsymbol{q}$ using the basis (4.9).

4.3.2 Application of POD on FMBD: component level

One can also implement POD on component level [5]. By separating the model into two or more components, a suitable basis can be determined for all components individually. All projections can be substituted into the constrained equation of motion, such that the system can be solved for $\boldsymbol{a}$ of all components. The advantage of application of POD on component level over system level is that, theoretically, the basis can be reduced further while finding solutions of the same accuracy compared to system level [5]. However, the reduction which can be achieved depends on the correlation between the generalized coordinates of the different components: when the correlation is low, the method will have the best result [5]. For example, points on one body are expected to have a high correlation compared to points on different bodies. So, defining the components as the different bodies in the system is a promising way to implement POD.

To apply POD on two components separately, one must first define which generalized coordinates correspond to which component. The coordinates belonging to the first and second components will be referred to as $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ respectively. A basis is determined for both sets separately, resulting in the projections:

$$\boldsymbol{q}_1 = \boldsymbol{U}_1\boldsymbol{a}_1 + \bar{\boldsymbol{q}}_1 \tag{4.15}$$

$$\boldsymbol{q}_2 = \boldsymbol{U}_2\boldsymbol{a}_2 + \bar{\boldsymbol{q}}_2 \tag{4.16}$$

where $\boldsymbol{U}_1$, $\boldsymbol{a}_1$ and $\bar{\boldsymbol{q}}_1$ are the basis, the vector of variables and the average of $\boldsymbol{q}_1$ of the first component respectively and $\boldsymbol{U}_2$, $\boldsymbol{a}_2$ and $\bar{\boldsymbol{q}}_2$ are the basis, the vector of variables and the average of $\boldsymbol{q}_2$ of the second component respectively [5]. Substituting the projections in the constrained equations of motion and projecting the top two rows into the transposed bases yields:

$$\begin{bmatrix} \boldsymbol{U}_1^T\boldsymbol{M}_1\boldsymbol{U}_1 & \boldsymbol{0} & \boldsymbol{U}_1^T{\boldsymbol{\Phi}_{\boldsymbol{q}_{1,1}}}^T & \boldsymbol{U}_1^T{\boldsymbol{\Phi}_{\boldsymbol{q}_{1,12}}}^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_2^T\boldsymbol{M}_2\boldsymbol{U}_2 & \boldsymbol{0} & \boldsymbol{U}_2^T{\boldsymbol{\Phi}_{\boldsymbol{q}_{2,12}}}^T & \boldsymbol{U}_2^T{\boldsymbol{\Phi}_{\boldsymbol{q}_{2,2}}}^T \\ \boldsymbol{\Phi}_{\boldsymbol{q}_{1,1}}\boldsymbol{U}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\Phi}_{\boldsymbol{q}_{1,12}}\boldsymbol{U}_1 & \boldsymbol{\Phi}_{\boldsymbol{q}_{2,12}}\boldsymbol{U}_2 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Phi}_{\boldsymbol{q}_{2,2}}\boldsymbol{U}_2 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{a}}_1 \\ \ddot{\boldsymbol{a}}_2 \\ \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_{12} \\ \boldsymbol{\lambda}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_1^T\boldsymbol{R}_1 \\ \boldsymbol{U}_2^T\boldsymbol{R}_2 \\ \boldsymbol{\gamma}_1 \\ \boldsymbol{\gamma}_{12} \\ \boldsymbol{\gamma}_2 \end{bmatrix} \tag{4.17}$$

where $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are the mass matrices of the first and second component respectively and

$$\boldsymbol{R}_1 = \boldsymbol{Q}_1 - \boldsymbol{K}_1(\boldsymbol{U}_1\boldsymbol{a}_1 + \bar{\boldsymbol{q}}_1) \tag{4.18}$$

$$\boldsymbol{R}_2 = \boldsymbol{Q}_2 - \boldsymbol{K}_2(\boldsymbol{U}_2\boldsymbol{a}_2 + \bar{\boldsymbol{q}}_2) \tag{4.19}$$

where $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are the applied forces on the first and second component respectively and $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are the stiffness matrices of the first and second component respectively [5]. The constraints are separated according to the components as well and denoted as $\boldsymbol{\Phi}_{\boldsymbol{q}_{s,t}}$, where t indicates on which component the constraint works and s indicates the coordinates to which the constraints are differentiated. So, $\boldsymbol{\Phi}_{\boldsymbol{q}_{s,1}}$ are the constraints working on component 1 only and correspond to the Lagrange multipliers $\boldsymbol{\lambda}_1$, $\boldsymbol{\Phi}_{\boldsymbol{q}_{s,2}}$ are the constraints working on component 2 only and correspond to $\boldsymbol{\lambda}_2$ and $\boldsymbol{\Phi}_{\boldsymbol{q}_{s,12}}$ are the constraints interacting between component 1 and 2 and correspond to the $\boldsymbol{\lambda}_{12}$.

By assembling the bases and the Jacobians of the constraints as:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_2 \end{bmatrix} \tag{4.20}$$

and:

$$\boldsymbol{\Phi}_{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{q}_{1,1}} & \mathbf{0} \\ \boldsymbol{\Phi}_{\boldsymbol{q}_{12,1}} & \boldsymbol{\Phi}_{\boldsymbol{q}_{2,12}} \\ \mathbf{0} & \boldsymbol{\Phi}_{\boldsymbol{q}_{2,2}} \end{bmatrix} \tag{4.21}$$

respectively, one can use (4.14) without expanding by hand to (4.17) [5].

Additionally to defining the different bodies as components, one can separate the rigid and flexible behaviour. The derivation of the reduced system of equations is the same, only the construction of $\boldsymbol{U}$ changes. Similar to the application of Craig Bampton reduction or free boundary modes, one can chose to solely discretize the flexible behaviour by defining the basis as:

$$\boldsymbol{U} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{U}_{1,flex} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{U}_{2,flex} \end{bmatrix} \tag{4.22}$$

where $\mathbf{1}$ are identity matrices with a number of rows and columns equal to the number of floating frame coordinates of each body, $\boldsymbol{U}_{1,flex}$ is the POD basis of the generalized flexible coordinates of the first component and $\boldsymbol{U}_{2,flex}$ is the POD basis of the generalized flexible coordinates of the second component. If the rigid behaviour is reduced as well, the basis becomes:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_{1,rig} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{U}_{1,flex} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{U}_{2,rig} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{U}_{2,flex} \end{bmatrix} \tag{4.23}$$

where $\boldsymbol{U}_{1,rig}$ is the POD basis of the floating frame coordinates of the first component and $\boldsymbol{U}_{2,rig}$ is the POD basis of the floating frame coordinates of the second component.

4.3.3 Obtaining and reducing the orthogonal basis

The quality of the result of POD reduced simulations strongly depends on the basis [4, 5, 15, 16]. Therefore, it is important to understand how the basis is obtained and what choices can be made in the construction and reduction of the basis.

Since the POD basis must represent the behaviour of the generalized coordinates, the basis is data dependent. Therefore, the basis is obtained using a snapshot matrix of the response of the system [4, 5, 15, 16]. The constrained equation of motion is evaluated over time, such that the vector of the generalized coordinates at m consecutive time instances is obtained and assembled into one matrix called the snapshot matrix:

$$\boldsymbol{S}_0 = \begin{bmatrix} \boldsymbol{q}(t_1) & \boldsymbol{q}(t_2) & \dots & \boldsymbol{q}(t_m) \end{bmatrix} \tag{4.24}$$

where $\boldsymbol{S}_0$ is the snapshot matrix and $\boldsymbol{q}(t_i)$ is the vector of generalized coordinates at time instance t_i. It is usual to collect at evenly distributed time steps [17]. One can determine the vector of average values of the generalized coordinates $\bar{\boldsymbol{q}}$ using the snapshots as well:

$$\bar{\boldsymbol{q}} = \frac{1}{m} \sum_{j=1}^{m} \boldsymbol{q}(t_j) \tag{4.25}$$

A zero mean ensemble is formed by subtracting the average coordinates (4.25) from the original snapshot matrix (4.24), obtaining the centralized snapshot matrix $\boldsymbol{S}$:

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{q}(t_1) - \bar{\boldsymbol{q}} & \boldsymbol{q}(t_2) - \bar{\boldsymbol{q}} & \dots & \boldsymbol{q}(t_m) - \bar{\boldsymbol{q}} \end{bmatrix} \tag{4.26}$$

To find the orthonormal basis which projects the original space onto the reduced order space, singular value decomposition (SVD) of the centralized snapshots is performed:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{4.27}$$

where $\boldsymbol{U}$ is a matrix containing the left-singular vectors, $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values and $\boldsymbol{V}$ is a matrix containing the right-singular vectors [4, 5, 15, 16]. The resulting matrix of left-singular vectors forms the POD basis. Alternatively, eigenvalue decomposition can be performed on the covariance matrix of the snapshot matrix $\boldsymbol{S}_0$:

$$\text{cov}(\boldsymbol{S}_0) = \boldsymbol{U}\sqrt{\boldsymbol{\Sigma}}\boldsymbol{U}^{-1} \tag{4.28}$$

where $\text{cov}(\boldsymbol{S}_0)$ is the covariance matrix of the non-centralized snapshot matrix, $\boldsymbol{U}$ is a matrix containing the eigenvectors and $\sqrt{\boldsymbol{\Sigma}}$ is a diagonal matrix containing the eigenvalues [4, 17]. The resulting eigenvectors can be used as basis as well, since both methods result in the same matrix $\boldsymbol{U}$. A derivation of the reason why both methods result in the same basis can be found in Appendix B.

Performing the projection using the complete basis does not result in a reduction of the constrained equation of motion, because the number of columns of $\boldsymbol{U}$ equals the number of generalized coordinates. If possible, the basis $\boldsymbol{U}$ can be truncated as:

$$\boldsymbol{U}_r = \boldsymbol{U}(:, 1:r) \tag{4.29}$$

where r is the truncation order and $\boldsymbol{U}_r$ is the truncated POD basis [4, 5]. The generalized coordinates are expressed in terms of the truncated bases as:

$$\boldsymbol{q}(t) \approx \boldsymbol{U}_r\boldsymbol{a}_r(t) + \bar{\boldsymbol{q}} \tag{4.30}$$

where $\boldsymbol{a}_r(t)$ is the vector of reduced variables [4, 5, 15, 16]. The vector of reduced variables $\boldsymbol{a}_r$ is consequentially only r elements long, reducing the number of unknown coefficients. By removing the last columns of the basis, the left-singular vectors corresponding to the lowest singular values are discarded, resulting in a basis $\boldsymbol{U}_r$ which consists only of the dominant singular vectors [4, 5]. When applying the truncation, an error is introduced since:

$$\boldsymbol{U}_r\boldsymbol{U}_r^T \approx \boldsymbol{1} \tag{4.31}$$

and:

$$\boldsymbol{S} \approx \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T \tag{4.32}$$

The error due to the truncation is defined as:

$$\epsilon_t = \sum_{k=r+1}^{N} \lambda_k^2 \Big/ \sum_{k=1}^{N} \lambda_k^2 \tag{4.33}$$

where λ_i are the singular values of the SVD of the centralized snapshot matrix. The error is a measure of the energy neglected in the approximation, because the singular values λ_i are a measure of the average kinematic energy captured in the snapshots [5]. Consequently, the error should be small.

If the reduction is not done on system but on component level, separate bases must be obtained for every component. A snapshot matrix must be constructed for the coordinates of each component separately and SVD should be performed separately on every snapshot matrix. The truncation order can be different for each basis.

4.4 The relation between POD and the natural vibration modes

To examine the relation between the basis used in POD and the natural vibration modes, one can examine the solution of the linearized equation of motion of an undamped and unforced multi degree of freedom system [18]. The equation of motion is written as:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} = \boldsymbol{0} \tag{4.34}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{q}$ is the vector of generalized coordinates and $\ddot{\boldsymbol{q}}$ is vector of accelerations. The solution of the equation of motion can be determined using the trial function:

$$\boldsymbol{q}(t) = \boldsymbol{u}e^{st} \tag{4.35}$$

where $\boldsymbol{u}$ is a vector of coefficients, s is a single coefficient and t is time. By substitution of the trail function into the equation of motion to obtain a solution for s, the response of the system can be written as:

$$\boldsymbol{q}(t) = \sum_{n=1}^{m} (A_n \cos(\omega_n t) + B_n \sin(\omega_n t))\boldsymbol{u}_n \tag{4.36}$$

where $\boldsymbol{u}$ are the eigenvectors of the system corresponding to the eigenvalues ω_n and A_n and B_n are coefficients depending on the initial conditions imposed on the system. The response (4.36) is written in vector form as:

$$\boldsymbol{q}(t) = \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_m \end{bmatrix} \begin{bmatrix} e_1(t) \\ \vdots \\ e_m(t) \end{bmatrix} \tag{4.37}$$

where

$$e_n(t) = A_n \cos(\omega_n t) + B_n \sin(\omega_n t) \tag{4.38}$$

To apply POD on the response of the system, one must first construct the snapshot matrix using the solution of the equation of motion (4.37):

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_m \end{bmatrix} \begin{bmatrix} e_1(t_1) & \cdots & e_1(t_p) \\ \vdots & \ddots & \vdots \\ e_m(t_1) & \cdots & e_m(t_p) \end{bmatrix} = \boldsymbol{U}\boldsymbol{E}^T \tag{4.39}$$

where p is the number of snapshots, q is the number of generalized coordinates, $\boldsymbol{U}$ is the matrix of size $q \times m$ containing all m eigenmodes and $\boldsymbol{E}$ is the matrix of size $p \times m$ containing all values of e_n for all snapshots. The snapshot matrix can be decoupled by writing

$$\boldsymbol{S} = \boldsymbol{U}\ [\mathbf{1}\ \mathbf{0}]\ [\boldsymbol{E}\ \boldsymbol{C}]^T \tag{4.40}$$

where $\mathbf{1}$ is an identity matrix of size $m \times m$, $\mathbf{0}$ is a matrix containing zeros of size $m \times (p - m)$ and $\boldsymbol{C}$ is a matrix of size $p \times (p - m)$, such that $\boldsymbol{C}$ does not affect the snapshot matrix because each entry of $\boldsymbol{C}$ is multiplied with zero [18]. One can recognise the shape of the decoupled snapshot matrix as the singular value decomposition, under the condition the matrices $\boldsymbol{U}$ and $[\boldsymbol{E}\ \boldsymbol{C}]^T$ are orthogonal. The matrices are orthogonal in case the mass matrix is proportional to the identity matrix $\mathbf{1}$ and the number of samples is infinite [18]. If the conditions are met, the POD basis of the system consists of the free vibration modes of the structure, showing the relation between the POD modes and the free boundary modes. However, one must recall that the constrained equations of motion are non-linear, meaning the relation between POD modes and free boundary modes is not directly applicable on the constrained equations of motion.

5 Model uncertainty

In the previous chapters, modelling of flexible multibody dynamic systems has been discussed under the assumption all input variables are exactly known. However, typically not all input parameters are known. One can easily imagine many geometric and material properties which are uncertain when modelling an existing structure or might show small deviations when building a modelled structure. The majority of the input parameters are naturally occurring properties which cannot have negative values and therefore have a log-normal distribution. The stochastic dynamic problem is obtained when including the uncertain input variables in the constrained equation of motion:

$$\begin{bmatrix} \boldsymbol{M}(\boldsymbol{\xi}) & \boldsymbol{\Phi_q}^T(\boldsymbol{q}(t,\boldsymbol{\xi}),\boldsymbol{\xi}) \\ \boldsymbol{\Phi_q}(\boldsymbol{q}(t,\boldsymbol{\xi}),\boldsymbol{\xi}) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}(t,\boldsymbol{\xi}) \\ \boldsymbol{\lambda}(t,\boldsymbol{\xi}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}(t,\boldsymbol{\xi}) - \boldsymbol{K}(\boldsymbol{\xi})\boldsymbol{q}(t,\boldsymbol{\xi}) \\ \boldsymbol{\gamma}(\boldsymbol{q}(t,\boldsymbol{\xi}),\dot{\boldsymbol{q}}(t,\boldsymbol{\xi}),\boldsymbol{\xi}) \end{bmatrix} \tag{5.1}$$

where $\boldsymbol{\xi}$ represents the uncertain variables and thus denotes the dependency of the parameters of the constrained equation of motion on the uncertain input variables. The stochastic constrained equation of motion can be used to determine the distributions of the output parameters.

A well-known method to obtain the distributions of the output parameters of a stochastic problem is Monte Carlo [3]. To perform Monte Carlo on the stochastic constrained equation of motion, one must sample the uncertain input parameters a large number of times and solve (5.1) for each sample [3]. The obtained set of solutions is used to approximate the distributions of the output variables. To achieve a good approximation of the distributions of the output parameters, including a good approximation of the tails of the distributions, a large number of samples is required. The number of samples required directly indicates the disadvantage of the Monte Carlo method: evaluating the constrained equation of motion over time is expansive, performing the evaluation a large number of times is therefore not feasible.

However, one can achieve reduction by discretizing the stochastic space. If the reduction is significant, applying Monte Carlo on the reduced system of equations is possible.

5.1 Discretization in the stochastic space

Discretization of the stochastic space can be achieved by use of shape functions to represent the stochastic space. Polynomial chaos expansion (PCE) can be used to achieve the discretization [6]. Note that discretizing the stochastic space by polynomials is comparable to FEM where the space is discretized by polynomials. Using PCE, a metamodel of the stochastic constrained equation of motion is obtained [6, 7, 8]. In the construction of the metamodel, it is assumed the different random input vectors are independent with known PDF's and the random output vector has finite variance [8]. The PC expansion of one of the generalized coordinates in the constrained equation of motion q at a specific time instance t_s is defined as:

$$q(t_s,\boldsymbol{\xi}) \approx \sum_{\boldsymbol{\alpha}\in\mathcal{A}} b_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \tag{5.2}$$

where $\boldsymbol{\xi}$ is a vector of the M different random input variables, $\boldsymbol{\alpha}$ is a multi-index, $b_{\boldsymbol{\alpha}}$ are the coefficients and $\Psi_{\boldsymbol{\alpha}}$ is the basis of multivariate polynomials of orders up to the truncation order p. The multivariate polynomials are constructed by a tensor product of univariate orthonormal polynomials $\Psi^{(i)}_{\alpha_i}(\xi_i)$ of order α_i [6, 7, 8, 19]:

$$\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \prod_{i=1}^{M} \Psi^{(i)}_{\alpha_i}(\xi_i) \tag{5.3}$$

such that one can write the expansion of q as:

$$q(t_s,\boldsymbol{\xi}) \approx \sum_{\boldsymbol{\alpha}\in\mathcal{A}} b_{\boldsymbol{\alpha}} \prod_{i=1}^{M} \Psi^{(i)}_{\alpha_i}(\xi_i) \tag{5.4}$$

For every multivariate polynomial in the basis, an ordered list $\boldsymbol{\alpha}$ of n-elements consisting of non-negative integers is defined:

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \tag{5.5}$$

of which the i^{th} element gives the order of the univariate polynomial $\Psi_{\alpha_i}^{(i)}(\xi_i)$. It can clearly be seen that the sum of the elements of $\boldsymbol{\alpha}$ determines the order of the resulting multivariate polynomial, leading to the condition:

$$0 \leq \sum_{i=1}^{M} \alpha_i = |\boldsymbol{\alpha}| \leq p \tag{5.6}$$

Graded lexicographic ordering will be used to define the multi-index [19]. Table 5.1 shows the multi-index in case of two uncertain input variables and a truncation order of three. Note that in the basis consists of ten ordered lists $\boldsymbol{\alpha}$, which means ten multivariate polynomials and ten coefficients are involved.

$\lvert\boldsymbol{\alpha}\rvert$	$\boldsymbol{\alpha}$
0	$(0,0)$
1	$(1,0)$
	$(0,1)$
2	$(2,0)$
	$(1,1)$
	$(0,2)$
3	$(3,0)$
	$(2,1)$
	$(1,2)$
	$(0,3)$

Table 5.1: Graded lexicographic ordering of a multi-index in case of two uncertain input parameters and a truncation order of three [19].

Which type of polynomial is used to construct the basis depends on the distribution of the random input variables [19, 7, 6]. Because the basis is orthonormal, the shape functions satisfy:

$$\mathbb{E}[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi})] = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} \tag{5.7}$$

where $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}$ is the Kronecker delta. If the univariate polynomials are orthonormal, the multivariate polynomial is as well. Therefore, the following condition must be met:

$$< \Psi_{\alpha_i}^{(i)}(\xi_i), \Psi_{\beta_i}^{(i)}(\xi_i) >= \int_{\xi \in \Xi} \Psi_{\alpha_i}^{(i)}(\xi_i)\Psi_{\beta_i}^{(i)}(\xi_i) f_{\xi_i}(\xi_i) d\xi_i = \delta_{\alpha_i \beta_i} \qquad i = 1, \ldots, M \tag{5.8}$$

where Ξ is the span of the stochastic space and the weight function $f_{\xi_i}(\xi_i)$ is the PDF of the random variable, showing the relation between the distribution of the random input variable and the polynomials in the basis: the weight function of the orthogonal polynomials $\Psi_{\alpha_i}^{(i)}(\xi_i)$ must equal the probability density function of the random variable ξ_i. An overview of which type of polynomial corresponds to which distribution is given in Table 5.2. As established, one would assign a log-normal distribution to most input parameters of the constrained equation of motion. Because the log-normal distribution is a transformation of the normal distribution, the Hermite polynomials normalized by the factorial of the order of the polynomials should be used in the bases [19].

Distribution type	**Polynomial type**	**Weight function**	**Interval variable**
Normal	Hermite $He_n(x)$	$e^{-x^2/2}$	$[-\infty, \infty]$
Uniform	Legendre $P_n(x)$	1	$[-1, 1]$
Beta	Jacobi $P_n^{(\alpha,\beta)}(x)$	$(1-x)^\alpha(1+x)^\beta$	$[-1, 1]$
Exponential	Laguerre $L_n(x)$	e^{-x}	$[0, \infty]$
Gamma	General Laguerre $L_n^{\alpha,\beta}(x)$	$x^\alpha e^{-x}$	$[0, \infty]$

Table 5.2: Distributions of random variables and the corresponding choice in polynomials [7, 19].

The expansion (5.4) can be constructed for each generalized coordinate, such that the solution of the constrained equation of motion (5.1) can be approximated by evaluating the metamodel instead. The

evaluation time of the expansions is very short compared to the evaluation time of the stochastic constrained equation of motion, because the expansions consist only of polynomials which are inexpensive to evaluate [6]. Because the evaluation time is short, application of Monte Carlo on the metamodels is possible. However, the expansions must approximate the stochastic constrained equation of motion closely for the application of Monte Carlo on the expansions to yield useful results. Expansions which approximate the constrained equation of motion well are achieved by tuning the coefficients $b_{\boldsymbol{\alpha}}$ such that the error between the PC expansions and the constrained equation of motion is minimal and by choosing a truncation order which can approximate the relation between the uncertain input parameters and the generalized coordinates well.

5.1.1 Optimization of the coefficients

The coefficients $b_{\boldsymbol{\alpha}}$ in the expansions of the generalized coordinates can be obtained through regression. One can generate a training data set of the generalized coordinates by sampling the distribution of the random input variables and evaluating the stochastic constrained equation of motion for each sample. By minimizing the squared error between the output of the expansions at the samples of the random input variables and the training data set, an optimal set of coefficients $b_{\boldsymbol{\alpha}}$ can be found [6]. The training data set must be large enough to tune all coefficients, but the use of too many data points will needlessly increase the computation time. The number of coefficients depends on the number of polynomials in the basis, which in turn depends on the truncation order p and the number of random input variables M. Typically, the number of sample points n required for the square error optimization is twice the number of coefficients:

$$n = 2\frac{(M+p)!}{M!+p!} \tag{5.9}$$

[6]. The squared error between the expansion and the generalized coordinate approximated by the expansion is defined as:

$$J(\boldsymbol{\xi_k}) = \Big(\sum_{\boldsymbol{\alpha}\in\mathcal{A}} b_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_k) - q(t_s, \boldsymbol{\xi}_k) \Big)^2 \tag{5.10}$$

where $\boldsymbol{\xi}_k$ is the vector of the k'th samples of the random input variables, J is the squared error and $q(t_s, \boldsymbol{\xi}_k)$ is the k'th element in the training set. To find the optimum set of coefficients, the error at every sample point in the training set should be minimized:

$$\hat{\boldsymbol{b}} = \arg\min \left(\left(\begin{bmatrix} \Psi_0(\boldsymbol{\xi}_1) & \Psi_1(\boldsymbol{\xi}_1) & \dots & \Psi_m(\boldsymbol{\xi}_1) \\ \Psi_0(\boldsymbol{\xi}_2) & \Psi_1(\boldsymbol{\xi}_2) & \dots & \Psi_m(\boldsymbol{\xi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_0(\boldsymbol{\xi}_n) & \Psi_1(\boldsymbol{\xi}_n) & \dots & \Psi_m(\boldsymbol{\xi}_n) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} - \begin{bmatrix} q(t_s, \boldsymbol{\xi}_1) \\ q(t_s, \boldsymbol{\xi}_2) \\ \vdots \\ q(t_s, \boldsymbol{\xi}_n) \end{bmatrix} \right)^2 \right) \tag{5.11}$$

where $\hat{\boldsymbol{b}}$ is the vector of optimized coefficients, n is the number of sample points in the training set and m is the number of multivariate polynomials in the basis. One can write (5.11) more compactly by defining the matrix of all multivariate polynomials evaluated at all sample points, the vector of coefficients and the training set of the generalized coordinates as:

$$\boldsymbol{A} = \begin{bmatrix} \Psi_0(\boldsymbol{\xi}_1) & \Psi_1(\boldsymbol{\xi}_1) & \dots & \Psi_m(\boldsymbol{\xi}_1) \\ \Psi_0(\boldsymbol{\xi}_2) & \Psi_1(\boldsymbol{\xi}_2) & \dots & \Psi_m(\boldsymbol{\xi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_0(\boldsymbol{\xi}_n) & \Psi_1(\boldsymbol{\xi}_n) & \dots & \Psi_m(\boldsymbol{\xi}_n) \end{bmatrix} \tag{5.12}$$

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{5.13}$$

$$\boldsymbol{q} = \begin{bmatrix} q(t_s, \boldsymbol{\xi}_1) \\ q(t_s, \boldsymbol{\xi}_2) \\ \vdots \\ q(t_s, \boldsymbol{\xi}_n) \end{bmatrix} \tag{5.14}$$

respectively. Therefore, (5.11) can be written as:

$$\hat{\boldsymbol{b}} = \arg\min([\boldsymbol{A}\boldsymbol{b} - \boldsymbol{q}]^2) \tag{5.15}$$

To solve the minimization problem (5.15), one must first rewrite the to be minimized expression:

$$\boldsymbol{J} = (\boldsymbol{A}\boldsymbol{b} - \boldsymbol{q})^2 \tag{5.16}$$

where $\boldsymbol{J}$ is a vector of the errors at all sample points, by expanding the brackets

$$\boldsymbol{J} = \boldsymbol{b}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{b} - 2\boldsymbol{q}^T\boldsymbol{A}\boldsymbol{b} + \boldsymbol{q}^T\boldsymbol{q} \tag{5.17}$$

The error is minimized at the sample points by equating the derivative of the error $\boldsymbol{J}$ with respect to the coefficients to zero:

$$\frac{\partial J}{\partial \boldsymbol{b}} = 2\hat{\boldsymbol{b}}^T\boldsymbol{A}^T\boldsymbol{A} - 2\boldsymbol{q}^T\boldsymbol{A} = 0 \tag{5.18}$$

which can be written as an expression for the optimized coefficients:

$$\hat{\boldsymbol{b}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{q} \tag{5.19}$$

Equation (5.19) can be used to find the optimum set of coefficients $\hat{\boldsymbol{b}}$ which minimizes the squared error at the training data set [6].

5.1.2 Tuning the truncation order

The truncation order determines the order of the relation between the input and output variables of the PC expansion. If the order is too low, the polynomials cannot estimate the relation between the generalized coordinate and the random variables well. However, when the order is too high the polynomials are overfitted on the training sample set [6], meaning the error of the metamodel on the training set is very small, but on points outside the training set the error might be very large. One can determine a proper truncation order by validating expansions of different truncation orders. The order which preforms best should be chosen.

To compare the expansions of different orders, one can use leave-one-out (LOO) cross validation [6]. To apply LOO cross validation, a sample set is generated which is one point larger compared the number of samples required for optimization of the coefficients (5.9) and thus consists of $n+1$ samples of each uncertain variable. All points but one are used as training set to optimize the coefficients. The remaining point is the test set and is used to determine the error between the expansion and the stochastic constrained equation of motion:

$$E = \frac{\left(q(t_s, \boldsymbol{\xi}_{test}) - \sum_{\boldsymbol{\alpha}\in\mathcal{A}} \hat{b}_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{test})\right)^2}{q(t_s, \boldsymbol{\xi}_{test})^2} \tag{5.20}$$

where $\boldsymbol{\xi}_{test}$ is the test set. By circling through the data set such that every point in the set is used as testing set once, the average LOO cross validation error is obtained. One can use the average error to compare the performance of expansions with different truncation orders.

6 Craig Bampton uncertainty

In the previous chapter, the effect of uncertain input variables on the output of the constrained equation of motion was considered. However, by taking one step back, an essential part of the simulations over time which might also be affected by uncertain input variables can be considered: the modes. The modes are used to simulate the flexible behaviour of the model and, in case of Craig Bampton, are dependent on the material properties. In case the material properties are uncertain, the stochastic space can be discretized to achieve reduction in the calculation of the Craig Bampton modes. The discretization can be achieved by applying POD in the stochastic space or by construction of a PC expansions of the Craig Bampton modes through stochastic Galerkin.

6.1 POD based Craig Bampton

The non-intrusive POD method can be used to achieve discretization of the stochastic space. Instead of applying the method in time as was done in 4.3, POD must be applied in the stochastic space. Performing the discretization by POD allows one to explore the relation between the POD modes and the boundary and internal modes according to Craig Bampton. A set of flexible modes shapes determined using Craig-Bampton always contains boundary modes, to which internal modes can be added in case a more detailed description of the behaviour of the internal nodes is required [1], as explained in section 4.2. The application of POD on Craig Bampton differs for the boundary and the internal modes and will thus be discussed separately.

6.1.1 Boundary modes

First, the application of POD on the boundary modes will be considered. For clarity, the deterministic expression for the boundary modes is repeated:

$$\boldsymbol{\Phi}_{CB} = \begin{bmatrix} \mathbf{1} \\ -\boldsymbol{K}_{ii}^{-1}\boldsymbol{K}_{ib} \end{bmatrix} \tag{6.1}$$

where $\boldsymbol{\Phi}_{CB}$ are the boundary modes, $\mathbf{1}$ is an identity matrix, $\boldsymbol{K}_{ii}$ is the section of the FE stiffness matrix representing the internal nodes and $\boldsymbol{K}_{ib}$ is the section of the FE stiffness matrix representing a coupling term between the internal and boundary nodes. Similar to the introduction of uncertainties to the constrained equation of motion, the stochastic expression for the boundary modes is written using the vector of uncertain input parameters $\boldsymbol{\xi}$:

$$\boldsymbol{\Phi}_{CB}(\boldsymbol{\xi}) = \begin{bmatrix} \mathbf{1} \\ -\boldsymbol{K}_{ii}^{-1}(\boldsymbol{\xi})\boldsymbol{K}_{ib}(\boldsymbol{\xi}) \end{bmatrix} \tag{6.2}$$

where $(\boldsymbol{\xi})$ denotes the dependency of the stiffness matrices and the boundary modes on the uncertain input variables. The identity matrix is not affected by the random variables, meaning the application of POD is only required on the second row of (6.2) which represent the deformation of the internal modes when the boundary modes are moved one unit in one direction. The stochastic response of the internal nodes is written more compactly by defining:

$$\boldsymbol{B}(\boldsymbol{\xi}) = -\boldsymbol{K}_{ii}^{-1}(\boldsymbol{\xi})\boldsymbol{K}_{ib}(\boldsymbol{\xi}) \tag{6.3}$$

where the s'th column of $\boldsymbol{B}$, denoted as $\boldsymbol{B}^s$, referrers to the s'th Craig Bampton boundary mode.

To achieve the discretization of the stochastic space, one can apply POD on each boundary mode. The method to obtain the POD basis has been explained in section 4.3. Because POD is applied in the stochastic space, the snapshot matrix contains m samples of the mode for different values of the random input variables instead of time:

$$\boldsymbol{S}^s = \begin{bmatrix} \boldsymbol{B}^s(\boldsymbol{\xi}_1) & \boldsymbol{B}^s(\boldsymbol{\xi}_2) & \cdots & \boldsymbol{B}^s(\boldsymbol{\xi}_m) \end{bmatrix} - \bar{\boldsymbol{B}}^s \qquad \forall s = 1, 2, 3, ..., n_{CB} \tag{6.4}$$

where $\boldsymbol{S}^s$ is the centralized snapshot matrix of the s'th boundary mode, $\boldsymbol{\xi}_i$ is the i'th set of samples of the random variables, n_{CB} is the number of boundary modes in the set of flexible mode shapes and:

$$\bar{\boldsymbol{B}}^s = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{B}^s(\boldsymbol{\xi}_i) \qquad \forall s = 1, 2, 3, ..., n_{CB} \tag{6.5}$$

One can obtain the POD bases of the different boundary modes by performing SVD on each centralized snapshot matrix:

$$\boldsymbol{S}^s = \boldsymbol{U}^s \boldsymbol{\Sigma}^s (\boldsymbol{V}^s)^T \qquad \forall s = 1, 2, 3, ..., n_{CB} \tag{6.6}$$

where $\boldsymbol{U}^s$, $\boldsymbol{\Sigma}^s$ and $\boldsymbol{V}^s$ are the matrix of left-singular vectors, the diagonal matrix containing the singular values and the matrix of right-singular vectors respectively, of the snapshot matrix of the s'th boundary mode. The matrices of left-singular vectors can be used as POD bases [4, 5, 15, 16].

Each bases can be reduced by disregarding the last columns of the bases:

$$\boldsymbol{U}_r^s = \boldsymbol{U}^s(:, 1:r) \qquad \forall s = 1, 2, 3, ..., n_{CB} \tag{6.7}$$

where $\boldsymbol{U}_r^s$ is the reduced basis for the s'th boundary mode and r is the truncation order. Ultimately, the reduced bases can be used to define the boundary modes as:

$$\boldsymbol{\Phi}_{CB}^s(\boldsymbol{\xi}) \approx \begin{bmatrix} \mathbf{1}^s \\ \boldsymbol{U}_r^s \boldsymbol{a}_r^s(\boldsymbol{\xi}) + \bar{\boldsymbol{B}}^s \end{bmatrix} \qquad \forall s = 1, 2, 3, ..., n_{CB} \tag{6.8}$$

where $\boldsymbol{\Phi}_{CB}^s$ denotes the s'th Craig Bampton boundary mode, $\mathbf{1}^s$ denotes the s'th column of the identity matrix from (6.2) and $\boldsymbol{a}_r^s$ is the vector of the reduced variables. An approximation of the boundary modes for any given sample of the uncertain parameters can be found by evaluating (6.8) to obtain the solution for the reduced variables.

6.1.2 Internal modes

Internal modes can be added to the set of boundary modes to allow for a more detailed simulation of the flexible behaviour of the internal nodes [1]. How the internal modes are defined is explained in section 4.2. The constrained eigenvalue problem is repeated for clarity:

$$(\boldsymbol{K}_{ii} - \lambda_s^2 \boldsymbol{M}_{ii}) \boldsymbol{\Phi}_{eig}^s = \mathbf{0} \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{6.9}$$

where $\boldsymbol{K}_{ii}$ is the section of the FE stiffness matrix representing the internal nodes, $\boldsymbol{M}_{ii}$ is the section of the FE mass matrix representing the internal nodes, $\boldsymbol{\Phi}_{eig}^s$ is the s'th internal mode corresponding to the s'th eigenfrequency λ_s and n_{eig} is the number of internal modes included in the set of flexible mode shapes. One can again define the stochastic problem by introducing the vector of uncertain input parameters $\boldsymbol{\xi}$ to the constrained eigenvalue problem:

$$\boldsymbol{K}_{ii}(\boldsymbol{\xi}) \boldsymbol{\Phi}_{eig}^s(\boldsymbol{\xi}) = (\lambda_s(\boldsymbol{\xi}))^2 \boldsymbol{M}_{ii} \boldsymbol{\Phi}_{eig}^s(\boldsymbol{\xi}) \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{6.10}$$

Application of POD on the internal modes to achieve discretization of the stochastic space involves the construction of a snapshot matrix containing m samples for each mode separately:

$$\boldsymbol{S}^s = \begin{bmatrix} \boldsymbol{\Phi}_{eig}^s(\boldsymbol{\xi}_1) & \boldsymbol{\Phi}_{eig}^s(\boldsymbol{\xi}_2) & \cdots & \boldsymbol{\Phi}_{eig}^s(\boldsymbol{\xi}_m) \end{bmatrix} - \bar{\boldsymbol{\Phi}}_{eig}^s \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{6.11}$$

where $\boldsymbol{S}^s$ is the centralized snapshot matrix of the s'th internal mode, $\boldsymbol{\xi}_i$ is the i'th set of samples of the uncertain variables and:

$$\bar{\boldsymbol{\Phi}}_{eig}^s = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\Phi}_{eig}^s(\boldsymbol{\xi}_i) \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{6.12}$$

When constructing the snapshot matrices, one should be careful to use the correct modes in the correct matrices. Multiple modes with the same eigenvalue are found of which the ordering can differ every time the problem is solved. Additionally, each mode can point either in the positive or negative direction. A proper snapshot matrix must contain one type of mode which all point in the same direction. The POD bases for all internal modes can be found by applying SVD on the centralized snapshot matrices:

$$\boldsymbol{S}^s = \boldsymbol{U}^s \boldsymbol{\Sigma}^s (\boldsymbol{V}^s)^T \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{6.13}$$

where $\boldsymbol{U}^s$, $\boldsymbol{\Sigma}^s$ and $\boldsymbol{V}^s$ are the matrix of left-singular vectors, the diagonal matrix containing the singular values and the matrix of right-singular vectors respectively of the snapshot matrix of the s'th internal mode. The POD bases are the matrices of left-singular vectors [4, 5, 15, 16].

As familiar, the bases can be reduced by excluding the last left-singular vectors according to (6.7). Using the reduced basis, one can approximate the s'th internal modes by:

$$\boldsymbol{\Phi}^s_{eig}(\xi) \approx \begin{bmatrix}\bar{\boldsymbol{\Phi}}^s_{eig} & \boldsymbol{U}^s_r\end{bmatrix}\begin{bmatrix}1\\ \boldsymbol{a}^s_r\end{bmatrix} = \hat{\boldsymbol{U}}^s_r\hat{\boldsymbol{a}}^s_r \qquad \forall s = 1,2,3,...,n_{eig} \tag{6.14}$$

where $\boldsymbol{a}^s_r$ is the vector of reduced variables, $\hat{\boldsymbol{U}}^s_r$ is a matrix of which the first row contains the centralizing term and the remaining rows contain the basis of the s'th internal mode and $\hat{\boldsymbol{a}}^s_r$ is a vector of which the first element equals one and the remaining elements are the reduced variables. By incorporating the centralizing term in the basis, the shape of the eigenvalue problem is maintained when the POD approximation (6.14) is substituted in the constrained eigenvalue problem (6.10). Performing the substitution and pre-multiplying the result by the transposed of the basis $\hat{\boldsymbol{U}}^s_r$ yields:

$$\Big[(\hat{\boldsymbol{U}}^s_r)^T\boldsymbol{K}_{ii}(\boldsymbol{\xi})\hat{\boldsymbol{U}}^s_r\Big]\hat{\boldsymbol{a}}^s_r(\boldsymbol{\xi}) = (\lambda^s(\boldsymbol{\xi}))^2\Big[(\hat{\boldsymbol{U}}^s_r)^T\boldsymbol{M}_{ii}\hat{\boldsymbol{U}}^s_r\Big]\hat{\boldsymbol{a}}^s_r(\boldsymbol{\xi}) \qquad \forall s = 1,2,3,...,n_{eig} \tag{6.15}$$

where the matrices $(\hat{\boldsymbol{U}}^s_r)^T\boldsymbol{K}_{ii}(\boldsymbol{\xi})\hat{\boldsymbol{U}}^s_r$ and $(\hat{\boldsymbol{U}}^s_r)^T\boldsymbol{M}_{ii}\hat{\boldsymbol{U}}^s_r$ are square, meaning (6.15) is an eigenvalue problem which can be solved for $\boldsymbol{a}^s_r$ for any given set of uncertain variables.

Because $\hat{\boldsymbol{U}}^s_r$ always contains at least two columns, the solution of the eigenvalue problem (6.15) will contain more than one eigenvector. When the basis is reduced to one (r=1), two equal eigenvalues are found. If a larger basis is used, additional higher eigenvalues are found. Practice shows the two equal eigenvalues are similar in value to the solution of the original problem (6.10). Because $\hat{\boldsymbol{a}}^s_r$ is expected to be one vector of which the first entry equals one and because any linear combination of modes in the same eigenspace is also an eigenvector, the sum of the two modes corresponding to the equal eigenvalues scaled such that the first entry equals one is used as the solution for $\hat{\boldsymbol{a}}^s_r$:

$$\hat{\boldsymbol{a}}^s = \frac{1}{\hat{\boldsymbol{a}}^s_{r,1}(1) + \hat{\boldsymbol{a}}^s_{r,2}(1)}(\hat{\boldsymbol{a}}^s_{r,1} + \hat{\boldsymbol{a}}^s_{r,2}) \qquad \forall s = 1,2,3,...,n_{eig} \tag{6.16}$$

where $\hat{\boldsymbol{a}}^s_{r,1}$ and $\hat{\boldsymbol{a}}^s_{r,2}$ are the modes corresponding to the two equal eigenvalues and the addition (1) indicates the first entry in the vectors.

6.2 Stochastic Galerkin on internal modes

Alternatively to POD, discretization of the stochastic space can be achieved by applying stochastic Galerkin on the mode shapes [9, 10]. POD is a non-intrusive method which requires a training set to obtain a good basis, but obtaining the data set and performing SVD both take time. Stochastic Galerkin is an intrusive method through which a PC expansion of the modes can be obtained without the use of a training set of the modes and might thus proof a good alternative to POD.

The set of flexible mode shapes obtained by Craig Bampton can contain both boundary and internal modes. The boundary modes are expected to be affected little by uncertain material properties, as long as the properties are continuous throughout the body. The internal modes and corresponding eigenvectors are expected to be influenced by uncertain material properties. Therefore, stochastic Galerkin will be applied on the internal modes only. The stochastic expression of the internal modes is repeated for convenience:

$$\boldsymbol{K}_{ii}(\boldsymbol{\xi})\boldsymbol{u}^s(\boldsymbol{\xi}) = \lambda^s(\boldsymbol{\xi})\boldsymbol{M}_{ii}\boldsymbol{u}^s(\boldsymbol{\xi}) \qquad \forall s = 1,2,3,...,n_{eig} \tag{6.17}$$

where $\boldsymbol{\xi}$ is the vector of uncertain input parameters, $\boldsymbol{K}_{ii}$ is the section of the FE stiffness matrix representing the internal nodes, $\boldsymbol{M}_{ii}$ is the section of the FE mass matrix representing the internal nodes, $\boldsymbol{u}^s_{eig}$ is the s'th internal mode corresponding to the s'th eigenfrequency λ_s and n_{eig} is the number of internal modes included in the set of flexible mode shapes.

6.2.1 Stochastic Galerkin pojection of the internal modes

The constrained eigenvalue problem (6.17) is of a generalized shape. A generalized eigenvalue problem can be rewritten to standard form using Cholesky factorization, which is beneficial because application of stochastic Galerkin on a standard eigenvalue problem is more straight forward [9, 10, 20]. Therefore, the discretization of the stochastic space for a standard eigenvalue problem is considered first. The stochastic standard eigenvalue problem is expressed as:

$$\boldsymbol{A}(\boldsymbol{\xi})\boldsymbol{u}^s(\boldsymbol{\xi}) = \lambda^s(\boldsymbol{\xi})\boldsymbol{u}^s(\boldsymbol{\xi}) \qquad \forall s = 1, 2, ..., n_s \tag{6.18}$$

where $\boldsymbol{A}$ is a stochastic matrix, $\boldsymbol{u}^s$ are the orthonormal eigenvectors corresponding to the eigenvalues λ^s and n_s is the number of eigenvalues and vectors considered.

The first step in the application of stochastic Galerkin on (6.18) is to define PC expansions for all variables [9, 10]. The expansion of matrix $\boldsymbol{A}$ is defined as:

$$\boldsymbol{A}(\boldsymbol{\xi}) = \sum_{\boldsymbol{\alpha}} \boldsymbol{A}^{(\boldsymbol{\alpha})} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \tag{6.19}$$

where $\boldsymbol{\alpha}$ is a multi-index, $\boldsymbol{A}^{(\boldsymbol{\alpha})}$ are the coefficients of the expansion and $\Psi_{\boldsymbol{\alpha}}$ are the multivariate polynomials. The expansion of the eigenvectors is given as:

$$\boldsymbol{u}^s(\boldsymbol{\xi}) = \sum_{\boldsymbol{\beta}} \boldsymbol{u}^{s,(\boldsymbol{\beta})} \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) \qquad \forall s = 1, 2, ..., n_s \tag{6.20}$$

where $\boldsymbol{\beta}$ is a multi-index, $\boldsymbol{u}^{s,(\boldsymbol{\beta})}$ are the coefficients of the expansion and $\Psi_{\boldsymbol{\beta}}$ are the multivariate polynomials. Finally, the expansion of the eigenvalues is:

$$\lambda^s(\boldsymbol{\xi}) = \sum_{\boldsymbol{\gamma}} \lambda^{s,(\boldsymbol{\gamma})} \Psi_{\boldsymbol{\gamma}}(\boldsymbol{\xi}) \qquad \forall s = 1, 2, ..., n_s \tag{6.21}$$

where $\boldsymbol{\gamma}$ is a multi-index, $\lambda^{s,(\boldsymbol{\gamma})}$ are the coefficients of the expansion and $\Psi_{\boldsymbol{\gamma}}$ are the multivariate polynomials. The expansions of the eigenvectors and values are of the same order n_ξ. Substitution of the expansions in the standard eigenvalue problem (6.18) yields:

$$\sum_{\boldsymbol{\alpha}} \boldsymbol{A}^{(\boldsymbol{\alpha})} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \sum_{\boldsymbol{\beta}} \boldsymbol{u}^{s,(\boldsymbol{\beta})} \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) = \sum_{\boldsymbol{\gamma}} \lambda^{s,(\boldsymbol{\gamma})} \Psi_{\boldsymbol{\gamma}}(\boldsymbol{\xi}) \sum_{\boldsymbol{\beta}} \boldsymbol{u}^{s,(\boldsymbol{\beta})} \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) \qquad \forall s = 1, 2, ..., n_s \tag{6.22}$$

If the polynomials are removed from (6.22), the result is a coupled deterministic system consisting of only the coefficients of the PC expansions [9, 10]. Because the coefficients of the uncertain input parameters are known, the deterministic system can be solved for the coefficients of the output variables, thus obtaining PC expansions of the output variables without the use of samples of the modes. To derive the deterministic system of coefficients, one must perform stochastic Galerkin projection on (6.22):

$$\Big\langle \sum_{\boldsymbol{\alpha}} \boldsymbol{A}^{(\boldsymbol{\alpha})} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \sum_{\boldsymbol{\beta}} \boldsymbol{u}^{s,(\boldsymbol{\beta})} \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}), \Psi_{\boldsymbol{k}}(\boldsymbol{\xi}) \Big\rangle = \Big\langle \sum_{\boldsymbol{\gamma}} \lambda^{s,(\boldsymbol{\gamma})} \Psi_{\boldsymbol{\gamma}}(\boldsymbol{\xi}) \sum_{\boldsymbol{\beta}} \boldsymbol{u}^{s,(\boldsymbol{\beta})} \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}), \Psi_{\boldsymbol{k}}(\boldsymbol{\xi}) \Big\rangle \qquad \forall s = 1, 2, ..., n_s \qquad \forall \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.23}$$

where $\boldsymbol{k}$ is a multi-index and $\Psi_{\boldsymbol{k}}$ is a set of multivariate polynomials [9, 10]. Equation (6.23) can be rewritten as:

$$\sum_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}} \boldsymbol{A}^{(\boldsymbol{\alpha})} \boldsymbol{u}^{s,(\boldsymbol{\beta})} \mathbb{E}\Big[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) \Psi_{\boldsymbol{k}}(\boldsymbol{\xi})\Big] = \sum_{\boldsymbol{\gamma}} \sum_{\boldsymbol{\beta}} \lambda^{s,(\boldsymbol{\gamma})} \boldsymbol{u}^{s,(\boldsymbol{\beta})} \mathbb{E}\Big[\Psi_{\boldsymbol{\gamma}}(\boldsymbol{\xi}) \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) \Psi_{\boldsymbol{k}}(\boldsymbol{\xi})\Big] \qquad \forall s = 1, 2, ..., n_s \qquad \forall \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.24}$$

The expectations can be evaluated knowing that the polynomials are orthonormal. The expectation of the product of two orthonormal polynomials is defined as:

$$\mathbb{E}[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi})] = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} \tag{6.25}$$

The product of two polynomials of the same type can be written as:

$$\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) = \sum_{\boldsymbol{\epsilon}=0}^{\boldsymbol{\alpha}+\boldsymbol{\beta}} c_{\boldsymbol{\alpha\beta}}^{(\boldsymbol{\epsilon})}\Psi_{\boldsymbol{\epsilon}}(\boldsymbol{\xi}) \tag{6.26}$$

where $c_{\boldsymbol{\alpha\beta}}^{(\boldsymbol{\epsilon})}$ is the delta tensor [21]. By combining (6.25) and (6.26), one can express the expectation of the product of three orthonormal polynomials as:

$$\mathbb{E}[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi})\Psi_{\boldsymbol{\gamma}}(\boldsymbol{\xi})] = c_{\boldsymbol{\beta\gamma}}^{(\boldsymbol{\alpha})} \tag{6.27}$$

[21]. A more detailed derivation and an efficient method to evaluate the delta tensor are shown in Appendix C. Substitution of (6.27) in the projection of the eigenvalue problem (6.24) yields:

$$\sum_{\boldsymbol{\alpha}}\sum_{\boldsymbol{\beta}} \boldsymbol{A}^{(\boldsymbol{\alpha})}\boldsymbol{u}^{s,(\boldsymbol{\beta})}c_{\boldsymbol{\beta k}}^{(\boldsymbol{\alpha})} = \sum_{\boldsymbol{\gamma}}\sum_{\boldsymbol{\beta}} \lambda^{s,(\boldsymbol{\gamma})}\boldsymbol{u}^{s,(\boldsymbol{\beta})}c_{\boldsymbol{\beta k}}^{(\boldsymbol{\gamma})} \qquad \forall s = 1,2,...,n_s \qquad \forall \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.28}$$

which is an expression of the standard eigenvalue problem from which all polynomials have been removed and thus contains solely the coefficients of the expansions. The summations over $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are removed by writing:

$$\sum_{\boldsymbol{\alpha}} \begin{bmatrix} c_{00}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi 0}^{(\boldsymbol{\alpha})} \\ \vdots & \ddots & \vdots \\ c_{0n_\xi}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi n_\xi}^{(\boldsymbol{\alpha})} \end{bmatrix} \otimes \boldsymbol{A}^{(\boldsymbol{\alpha})} \begin{bmatrix} \boldsymbol{u}^{s,(0)} \\ \vdots \\ \boldsymbol{u}^{s,(n_\xi)} \end{bmatrix} = \sum_{\boldsymbol{\gamma}} \begin{bmatrix} c_{00}^{(\boldsymbol{\gamma})} & \cdots & c_{n_\xi 0}^{(\boldsymbol{\gamma})} \\ \vdots & \ddots & \vdots \\ c_{0n_\xi}^{(\boldsymbol{\gamma})} & \cdots & c_{n_\xi n_\xi}^{(\boldsymbol{\gamma})} \end{bmatrix} \otimes (\lambda^{s,(\boldsymbol{\gamma})}\boldsymbol{I_A}) \begin{bmatrix} \boldsymbol{u}^{s,(0)} \\ \vdots \\ \boldsymbol{u}^{s,(n_\xi)} \end{bmatrix} \qquad \forall s = 1,2,...,n_s \tag{6.29}$$

where $\boldsymbol{I_A}$ is the identity matrix of the same size as $\boldsymbol{A}(\xi)$. By defining the matrix of delta tensors and the vector of coefficients of the eigenvectors as:

$$\boldsymbol{\Delta}^{(\boldsymbol{\alpha})} = \begin{bmatrix} c_{00}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi 0}^{(\boldsymbol{\alpha})} \\ \vdots & \ddots & \vdots \\ c_{0n_\xi}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi n_\xi}^{(\boldsymbol{\alpha})} \end{bmatrix} \tag{6.30}$$

$$\bar{\boldsymbol{u}}^s = \begin{bmatrix} \boldsymbol{u}^{s,(0)} \\ \vdots \\ \boldsymbol{u}^{s,(n_\xi)} \end{bmatrix} \tag{6.31}$$

respectively, one can write (6.29) more compactly:

$$\Big(\sum_{\boldsymbol{\alpha}} \boldsymbol{\Delta}^{(\boldsymbol{\alpha})} \otimes \boldsymbol{A}^{(\boldsymbol{\alpha})}\Big)\bar{\boldsymbol{u}}^s = \Big(\sum_{\boldsymbol{\gamma}} \boldsymbol{\Delta}^{(\boldsymbol{\gamma})} \otimes \lambda^{s,(\boldsymbol{\gamma})}\boldsymbol{I_A}\Big)\bar{\boldsymbol{u}}^s \qquad \forall s = 1,2,...,n_s \tag{6.32}$$

The obtained system (6.32) is the deterministic system containing only the coefficients of the expansions of the input and output parameters of the standard eigenvalue problem (6.18). Note that the system has maintained the shape of a standard eigenvalue problem and can thus be solved for the coefficients of the eigenvalues and vectors if the expansion of matrix $\boldsymbol{A}$ is known.

The orthonormality condition is imposed on the eigenvectors of the standard eigenvalue problem, leading to a condition imposed on the coefficients of the eigenvectors as well. The condition can be attained by projecting the orthonormality condition onto $\Psi_{\boldsymbol{k}}(\xi)$:

$$\Big\langle \boldsymbol{u}^s(\boldsymbol{\xi})^T\boldsymbol{u}^s(\boldsymbol{\xi}), \Psi_{\boldsymbol{k}}(\boldsymbol{\xi})\Big\rangle = \delta_{\boldsymbol{k}1} \qquad \forall s = 1,2,...,n_s \qquad \forall \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.33}$$

where $\delta_{\boldsymbol{k}1}$ is the delta tensor [9]. Substitution of the expansions of the eigenvectors (6.20) in the projection (6.33) and expansion of the expected values using (6.27) yields:

$$\sum_{\boldsymbol{i}}\sum_{\boldsymbol{j}} \boldsymbol{u}^{s,(\boldsymbol{i})^T}\boldsymbol{u}^{s,(\boldsymbol{j})}c_{\boldsymbol{ij}}^{(\boldsymbol{k})} = \delta_{\boldsymbol{k}1} \qquad \forall s = 1,2,...,n_s \qquad \forall \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.34}$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ are multi-indices. One can write (6.34) in matrix form:

$$\sum_{\boldsymbol{i}}\sum_{\boldsymbol{j}}\begin{bmatrix} c_{00}^{(\boldsymbol{k})}\boldsymbol{u}^{s,(0)^T}\boldsymbol{u}^{s,(0)} & \dots & c_{0n_\xi}^{(\boldsymbol{k})}\boldsymbol{u}^{s,(0)^T}\boldsymbol{u}^{s,(n_\xi)} \\ \vdots & \ddots & \vdots \\ c_{n_\xi 0}^{(\boldsymbol{k})}\boldsymbol{u}^{s,(n_\xi)^T}\boldsymbol{u}^{s,(0)} & \dots & c_{n_\xi n_\xi}^{(\boldsymbol{k})}\boldsymbol{u}^{s,(n_\xi)^T}\boldsymbol{u}^{s,(n_\xi)} \end{bmatrix}_{\boldsymbol{ij}} = \delta_{\boldsymbol{k}1} \qquad \forall s = 1,2,...,n_s \qquad \forall \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.35}$$

which can be written more compactly as:

$$\sum_i \sum_j \left[\boldsymbol{\Delta}^{(k)} \circ \left\langle \boldsymbol{u}^{s,(i)}, \boldsymbol{u}^{s,(j)} \right\rangle\right]_{ij} = \delta_{k1} \qquad \forall s = 1,2,...,n_s \qquad \forall \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.36}$$

where $\circ$ denotes the Hadamard product. Equation (6.36) is the deterministic system containing only the coefficients which defines the orthonormality condition imposed on the eigenvectors.

The two deterministic systems (6.32) and (6.36) describe the standard eigenvalue problem in terms of the coefficients of the PC expansions only. $\boldsymbol{A}(\xi)$ is an input parameter of which the coefficients are known and the elements of the delta tensor can be easily attained as well. The only unknown is the system are the coefficient of the eigenvalues and modes and can thus be found by solving the systems. However, the constrained eigenvalue problem which defines the Craig Bampton internal modes is not a standard but a generalized eigenvalue problem. Nonetheless, the derived deterministic systems (6.32) and (6.36) can still be used, because the generalized problem can be written in the standard form by application of Cholesky factorization on the internal mass matrix:

$$\boldsymbol{M}_{ii} = \boldsymbol{L}\boldsymbol{L}^T \tag{6.37}$$

where $\boldsymbol{L}$ is the lower triangular matrix [9, 10, 20]. By substitution of the factorization of the internal mass matrix in the constrained equation of motion (6.17) and pre-multiplication of the expression with the inverse of $\boldsymbol{L}$, one obtains:

$$(\boldsymbol{L}^{-1}\boldsymbol{K}_{ii}(\boldsymbol{\xi})\boldsymbol{L}^{-T})(\boldsymbol{L}^T\boldsymbol{u}^s(\boldsymbol{\xi})) = \lambda^s(\boldsymbol{L}^T\boldsymbol{u}^s(\boldsymbol{\xi})) \tag{6.38}$$

The resulting expression (6.38) has the same shape as the standard eigenvalue problem in which matrix $\boldsymbol{A}$ is given as:

$$\boldsymbol{A}(\boldsymbol{\xi}) = \boldsymbol{L}^{-1}\boldsymbol{K}_{ii}(\boldsymbol{\xi})\boldsymbol{L}^{-T} \tag{6.39}$$

and the eigenvectors are expressed as:

$$\boldsymbol{w}^s(\boldsymbol{\xi}) = \boldsymbol{L}^T\boldsymbol{u}^s(\boldsymbol{\xi}) \tag{6.40}$$

The coefficients of matrix $\boldsymbol{A}$ can be calculated using the coefficients of the internal stiffness matrix:

$$\boldsymbol{A}(\boldsymbol{\xi}) = \sum_\alpha \boldsymbol{L}^{-1}\boldsymbol{K}_{ii}^{(\boldsymbol{\alpha})}\boldsymbol{L}^{-T}\Psi_\alpha(\boldsymbol{\xi}) \tag{6.41}$$

So, by applying the redefinition of $\boldsymbol{A}(\xi)$ (6.39) and the eigenvectors (6.40), the deterministic systems (6.32) and (6.36) can be used to solve the PC coefficients of the eigenvalues and modes of the constrained eigenvalue problem.

Because (6.32) has the shape of an eigenvalue problem, the coefficients of the eigenvalues can in principle be found by solving the system. However, the expression containing the coefficients of the eigenvalues is complicated. Instead, an easier to evaluate expression for the coefficients of the eigenvalues is found by pre-multiplication of the standard eigenvalue problem (6.18) with the transposed of the modes:

$$\lambda^s(\boldsymbol{\xi}) = (\boldsymbol{u}^s(\boldsymbol{\xi}))^T\boldsymbol{A}(\boldsymbol{\xi})\boldsymbol{u}^s(\boldsymbol{\xi}) \qquad \forall s = 1,2,...,n_s \tag{6.42}$$

To find an expression for the coefficients of the eigenvalues, (6.42) must be projected onto $\Psi_{\boldsymbol{k}}(\boldsymbol{\xi})$:

$$\lambda^{s,(k)} = \left\langle (\boldsymbol{u}^s(\boldsymbol{\xi}))^T\boldsymbol{A}(\boldsymbol{\xi})\boldsymbol{u}^s(\boldsymbol{\xi}), \Psi_{\boldsymbol{k}}(\boldsymbol{\xi}) \right\rangle \qquad \forall s = 1,2,...,n_s \qquad \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.43}$$

One can define a vector $\boldsymbol{v}(\boldsymbol{\xi})$:

$$\boldsymbol{v}^s(\boldsymbol{\xi}) = \boldsymbol{A}(\boldsymbol{\xi})\boldsymbol{u}^s(\boldsymbol{\xi}) \qquad \forall s = 1,2,...,n_s \qquad \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.44}$$

of which the coefficients can be expressed in terms of the coefficients of $\boldsymbol{A}$ and $\boldsymbol{u}^s$ by applying stochastic Galerkin projection:

$$\boldsymbol{v}^{s,(\boldsymbol{k})} = \Big\langle \boldsymbol{A}(\boldsymbol{\xi})\boldsymbol{u}^s(\boldsymbol{\xi}), \Psi_{\boldsymbol{k}}(\boldsymbol{\xi}) \Big\rangle \qquad \forall s = 1, 2, ..., n_s \qquad \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.45}$$

The projection (6.45) can be expanded to:

$$\boldsymbol{v}^{s,(\boldsymbol{k})} = \Big(\sum_{\boldsymbol{\alpha}} \begin{bmatrix} c_{0\boldsymbol{k}}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi \boldsymbol{k}}^{(\boldsymbol{\alpha})} \end{bmatrix} \otimes \boldsymbol{A}^{(\boldsymbol{\alpha})} \Big) \bar{\boldsymbol{u}}^s \qquad \forall s = 1, 2, ..., n_s \qquad \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.46}$$

The vector $\boldsymbol{v}(\boldsymbol{\xi})$ can be substituted in the expression for the coefficients of the eigenvalues (6.43) and expanded using (6.46) to obtain:

$$\lambda^{s,(\boldsymbol{k})} = \sum_{\boldsymbol{i}} \sum_{\boldsymbol{j}} \Big[\boldsymbol{\Delta}^{(\boldsymbol{k})} \circ \Big\langle \boldsymbol{u}^{s,(\boldsymbol{i})}, \boldsymbol{v}^{s,(\boldsymbol{j})} \Big\rangle \Big]_{\boldsymbol{ij}} \qquad \forall s = 1, 2, ..., n_s \qquad \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.47}$$

of which the derivation is not shown explicitly because of the similarity to the derivation (6.33) to (6.36). The derivated relation 6.47 can be used to calculate the coefficients of the eigenvalues after the coefficients of the eigenmodes have been solved using (6.32) [9]. For sake of clarity on how (6.47) should be evaluated, the full expression is written here:

$$\sum_{\boldsymbol{i}} \sum_{\boldsymbol{j}} \begin{bmatrix} c_{00}^{(\boldsymbol{k})} \boldsymbol{u}^{\boldsymbol{k},(0)^T} \Big(\sum_{\boldsymbol{\alpha}} \begin{bmatrix} c_{00}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi 0}^{(\boldsymbol{\alpha})} \end{bmatrix} \otimes \boldsymbol{A}^{(\boldsymbol{\alpha})} \Big) \bar{\boldsymbol{u}}^s & \cdots & c_{0 n_\xi}^{(\boldsymbol{k})} \boldsymbol{u}^{\boldsymbol{k},(0)^T} \Big(\sum_{\boldsymbol{\alpha}} \begin{bmatrix} c_{0 n_\xi}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi n_\xi}^{(\boldsymbol{\alpha})} \end{bmatrix} \otimes \boldsymbol{A}^{(\boldsymbol{\alpha})} \Big) \bar{\boldsymbol{u}}^s \\ \vdots & \ddots & \vdots \\ c_{n_\xi 0}^{(\boldsymbol{k})} \boldsymbol{u}^{\boldsymbol{k},(n_\xi)^T} \Big(\sum_{\boldsymbol{\alpha}} \begin{bmatrix} c_{00}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi 0}^{(\boldsymbol{\alpha})} \end{bmatrix} \otimes \boldsymbol{A}^{(\boldsymbol{\alpha})} \Big) \bar{\boldsymbol{u}}^s & \cdots & c_{n_\xi n_\xi}^{(\boldsymbol{k})} \boldsymbol{u}^{\boldsymbol{k},(n_\xi)^T} \Big(\sum_{\boldsymbol{\alpha}} \begin{bmatrix} c_{0 n_\xi}^{(\boldsymbol{\alpha})} & \cdots & c_{n_\xi n_\xi}^{(\boldsymbol{\alpha})} \end{bmatrix} \otimes \boldsymbol{A}^{(\boldsymbol{\alpha})} \Big) \bar{\boldsymbol{u}}^s \end{bmatrix}_{\boldsymbol{ij}}$$
$$\forall s = 1, 2, ..., n_s \qquad \boldsymbol{k} = \boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_{n_\xi} \tag{6.48}$$

In conclusion, three deterministic systems (6.32), (6.36) and (6.47) are obtained which consist solely of the known coefficients of the expansions of the internal stiffness matrix, the Cholesky factorisation of the internal mass matrix and the unknown coefficients of the expansions of the eigenvalues and internal modes and can thus be used to solve the coefficients of the eigenvalues and internal modes.

6.2.2 Solution methods

The deterministic systems of coefficients found in the previous section can be solved to obtain the coefficients of the expansions of the constrained eigenvalues and internal modes if the coefficients of the expansion of the stiffness matrix are known. If the uncertain variable(s) and the stiffness matrix are linearly related, the coefficients of the stiffness matrix can be obtained directly using the characteristics of the distributions of the uncertain variables [22], as explained in detail in Appendix D. In case the uncertain variable(s) and the stiffness matrix are not linearly related, the coefficients can be obtained by minimization of the square error between the expansion and the original formulation of the stiffness matrix on a set of training points [6]. The regression method has been explained in detail in section 5.1.1.

When the coefficients of the expansion of the stiffness matrix are known, one can solve the deterministic systems for the coefficients of the output parameters. The problem of solving the systems can be approached in multiple ways. The first solution method which is discussed will be called the straight forward method and involves, as the name suggests, solving the deterministic eigenvalue problem (6.32) directly. The difficulty which raises is that the vector of coefficients of the eigenvectors consists of the coefficients of all orders 0 to n_ξ of the modes:

$$\bar{\boldsymbol{u}}^s = \begin{bmatrix} \boldsymbol{u}^{s,(0)} \\ \vdots \\ \boldsymbol{u}^{s,(n_\xi)} \end{bmatrix} \tag{6.49}$$

Solving the eigenvalue problem ensures orthogonality of the coefficients for different modes s, but the condition (6.36) requires the vectors of coefficients of different orders k of the same mode s ($\boldsymbol{u}^{s,(k)}$) to be

orthonormal as well. One can assure the orthonormality condition by, after solving the eigenvalue problem, dividing every set of coefficients $\boldsymbol{u}^{s,(k)}$ by its length and applying Gram-Schmidt on the result [9, 10]. Gram-Schmidt is a well-known method to orthonormalize vectors and will be explained in more detail in section 6.2.3. Finally, the obtained coefficients of the eigenvectors can be substituted in (6.47) to find the coefficients of the eigenvalues. Algorithm 2 shows an overview of the straight forward method.

The matrix on which the eigenvalue decomposition is done in the straight forward method is the outer product of the coefficients of $\boldsymbol{A}(\xi)$ and the delta tensor and is thus p_β times as large as $\boldsymbol{A}(\xi)$, adding dimensions to the problem. Due to the additional dimensions, the resulting eigenvalues repeat the eigenvalues of the original problem p_β times. For example, for a three-dimensional beam, one of the Craig Bampton internal modes has the shape of halve a sin between boundary modes. Because the middle node of the beam is fixed and the sin shape can occur along two axes, the one mode shape will appear four times. Because the shapes are all the same and the material properties are assumed to be constant throughout the beam, the same eigenvalue belongs to each of the halve-sin-shaped modes and will thus be found four times. When applying the straight forward method on expansions for the modes and eigenvalues including orders zero and one, the eigenvalue belonging to the halve-sin-shaped modes will be found eight times instead of four. The repetition of eigenmodes and values due to the added dimensions must be considered when comparing results.

Algorithm 2: The straight forward method to find the PCE coefficients of the modes and eigenfrequencies.

Input : $p_{\boldsymbol{\alpha}}$, $p_{\boldsymbol{\beta}}$, $p_{\boldsymbol{\gamma}}$, μ and σ of the uncertain variables $\boldsymbol{x}$
Output: $u^{s,(\boldsymbol{\beta})}$, $\lambda^{s,(\boldsymbol{\gamma})}$

1 Determine the coefficients $\boldsymbol{K}_{ii}^{(\boldsymbol{\alpha})}$:
- Direct: Find the coefficients of the uncertain variables $\boldsymbol{x}(\boldsymbol{\xi})$: $\boldsymbol{x}^{(\boldsymbol{i})} = \mathbb{E}[\boldsymbol{x}(\boldsymbol{\xi})\Psi_{\boldsymbol{i}}(\boldsymbol{\xi})]$ for all x and $\boldsymbol{i}$
- Find the coefficients of $\boldsymbol{K}_{ii}(\boldsymbol{\xi})$: $\boldsymbol{K}^{(\boldsymbol{\alpha})} = \boldsymbol{K}(\boldsymbol{x}^{(\boldsymbol{\alpha})})$
- Sampling: Take n samples $\boldsymbol{\xi}$ and assemble: $\boldsymbol{\Psi} = [\Psi_0(\boldsymbol{\xi}) \ \dots \ \Psi_{\boldsymbol{\alpha}_{(p_{\boldsymbol{\alpha}})}}(\boldsymbol{\xi})]$
- Minimize the error on the samples: $\bar{\boldsymbol{K}} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{K}(\boldsymbol{x}(\boldsymbol{\xi}))$;

2 Perform Cholesky factorization on $\boldsymbol{M}_{ii}$: $\boldsymbol{L} = \text{chol}(\boldsymbol{M}_{ii}, \text{'lower'})$;

3 Solve the eigenvalue problem: $[\boldsymbol{V}, \boldsymbol{D}] = \text{eig}(\sum_{\boldsymbol{\alpha}} \boldsymbol{\Delta}^{(\boldsymbol{\alpha})} \otimes (\boldsymbol{L}^{-1}\boldsymbol{K}_{ii}^{(\boldsymbol{\alpha})}\boldsymbol{L}^{-\boldsymbol{T}}))$;

4 Normalise the coefficients of the modes $\boldsymbol{u}^{s,(\boldsymbol{\beta})}$: $\boldsymbol{u}^{s,(\boldsymbol{\beta})} = \frac{\boldsymbol{u}^{s,(\boldsymbol{\beta})}}{\sqrt{\sum(\boldsymbol{u}^{s,(\boldsymbol{\beta})})^2}}$;

5 Orthonormalize $\boldsymbol{u}^{s,(\boldsymbol{\beta})}$ by Gram-Schmidt: $\boldsymbol{u}^{s,(\boldsymbol{\beta})} = \boldsymbol{v}^{s,(\boldsymbol{\beta})} - \sum_{j=0}^{\boldsymbol{\beta}-1} \frac{\langle \boldsymbol{v}^{s,(\boldsymbol{\beta})}, \boldsymbol{u}^{s,(\boldsymbol{j})} \rangle}{\langle \boldsymbol{u}^{s,(\boldsymbol{j})}, \boldsymbol{u}^{s,(\boldsymbol{j})} \rangle} \boldsymbol{u}^{s,(\boldsymbol{j})}$;

6 Calculate the coefficients of the eigenvalues: $\lambda^{s,(\boldsymbol{\gamma})} = \sum_{\boldsymbol{i}} \sum_j \left[\boldsymbol{\Delta}^{(\boldsymbol{\gamma})} \circ \left\langle \boldsymbol{u}^{s,(\boldsymbol{i})}, \boldsymbol{v}^{s,(\boldsymbol{j})} \right\rangle \right]_{\boldsymbol{ij}}$;

Alternatively to the straight forward method, the deterministic systems can be solved using the inexact method. Instead of using eigenvalue decomposition to solve the complete system (6.32), the inexact method requires the eigenvalue problem to only be solved for the zeroth order coefficients [9, 10]. All higher order coefficients are initially assumed to be zero [9, 10]. To obtain estimations of the higher order coefficients, the deterministic system of the eigenvalue problem (6.32) is re-approached as:

$$\bar{\boldsymbol{u}}^s = (\sum_{\boldsymbol{\alpha}} \boldsymbol{\Delta}^{(\boldsymbol{\alpha})} \otimes (\boldsymbol{L}^{-1}\boldsymbol{K}_{ii}^{(\boldsymbol{\alpha})}\boldsymbol{L}^{-\boldsymbol{T}}))\bar{\boldsymbol{v}}^s \tag{6.50}$$

and iteratively applied to obtain good estimations of the higher order coefficients of the internal modes by substitution of the initial guess for $\bar{\boldsymbol{u}}^s$ and solving the system for $\bar{\boldsymbol{v}}^s$ using conjugate gradients [9, 10]. The resulting coefficients of the eigenvalues do not meet the orthonormality condition (6.36) however. Therefore, the Gram-Schmidt method must be applied to ensure orthonormality [9, 10]. After obtaining the coefficients for the eigenvalues, one can substitute the coefficients in (6.47) to find the coefficients of the eigenvalues. Algorithm 3 gives an overview of the inexact method.

Unlike in the straight forward method, the eigenvalue decomposition in the inexact method is applied on a matrix of the same dimensions as the original problem. Therefore, the initial guess for $\bar{\boldsymbol{u}}^{s,(0)}$ does not contain additional repetitions of eigenmodes and neither will the eigenvalues.

Algorithm 3: The inexact method to find the PCE coefficients of the modes and eigenfrequencies. [9, 10]

Input : $p_{\boldsymbol{\alpha}}$, $p_{\boldsymbol{\beta}}$, $p_{\boldsymbol{\gamma}}$, μ and σ of the uncertain variables $\boldsymbol{x}$
Output: $u^{s,(\beta)}$, $\lambda^{s,(\gamma)}$

Determine the coefficients $\boldsymbol{K}_{ii}^{(\boldsymbol{\alpha})}$:
- Direct: Find the coefficients of the uncertain variables $\boldsymbol{x}(\boldsymbol{\xi})$: $\boldsymbol{x}^{(\boldsymbol{i})} = \mathbb{E}[\boldsymbol{x}(\boldsymbol{\xi})\Psi_{\boldsymbol{i}}(\boldsymbol{\xi})]$ for all x and $\boldsymbol{i}$
- Find the coefficients of $\boldsymbol{K}_{ii}(\boldsymbol{\xi})$: $\boldsymbol{K}^{(\boldsymbol{\alpha})} = \boldsymbol{K}(\boldsymbol{x}^{(\boldsymbol{\alpha})})$
- Sampling: Take n samples $\boldsymbol{\xi}$ and assemble: $\boldsymbol{\Psi} = [\Psi_0(\boldsymbol{\xi}) \ \dots \ \Psi_{\boldsymbol{\alpha}_{p_{(\boldsymbol{\alpha})}}}(\boldsymbol{\xi})]$
- Minimize the error on the samples: $\bar{\boldsymbol{K}} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{K}(\boldsymbol{x}(\boldsymbol{\xi}))$;

Perform Cholesky factorization on $\boldsymbol{M}_{ii}$: $\boldsymbol{L} = \text{chol}(\boldsymbol{M}_{ii}, \text{'lower'})$;

Solve the eigenvalue problem for the zeroth order coefficients:
$$[\boldsymbol{V}, \boldsymbol{D}] = \text{eig}(\boldsymbol{\Delta}^{(0)} \otimes (\boldsymbol{L}^{-1}\boldsymbol{K}_{ii}^{(0)}\boldsymbol{L}^{-\boldsymbol{T}})) \ ;$$

Determine and normalize the coefficients of the modes $\boldsymbol{u}^{s,(\beta)}$:

Initialize: $\boldsymbol{u}^{s,(0)} = \boldsymbol{v}^s$, $\boldsymbol{u}^{s,(i)} = \boldsymbol{0}$ for $s = 1, 2, ..., n_s$ and $i = 1, 2, ..., p_\beta$

for $s = 0, 1, ..., n_s$ **do**
- By conjugate gradients solve: $\bar{\boldsymbol{w}}^s = \text{pcg}(\sum_{\boldsymbol{\alpha}} \boldsymbol{\Delta}^{(\boldsymbol{\alpha})} \otimes (\boldsymbol{L}^{-1}\boldsymbol{K}_{ii}^{(\boldsymbol{\alpha})}\boldsymbol{L}^{-\boldsymbol{T}}), \bar{\boldsymbol{u}}^s)$
- Orthonormalize $\boldsymbol{u}^{s,(\boldsymbol{\beta})}$ by Gram-Schmidt: $\boldsymbol{u}^{s,(\boldsymbol{\beta})} = \boldsymbol{w}^{s,(\boldsymbol{\beta})} - \sum_{j=0}^{\boldsymbol{\beta}-1} \frac{\langle \boldsymbol{w}^{s,(\boldsymbol{\beta})}, \boldsymbol{u}^{s,(\boldsymbol{j})} \rangle}{\langle \boldsymbol{u}^{s,(\boldsymbol{j})}, \boldsymbol{u}^{s,(\boldsymbol{j})} \rangle} \boldsymbol{u}^{s,(\boldsymbol{j})}$
- Check convergence

end

Calculate the coefficients of the eigenvalues: $\lambda^{s,(\boldsymbol{\gamma})} = \sum_{\boldsymbol{i}} \sum_{\boldsymbol{j}} \left[\boldsymbol{\Delta}^{(\boldsymbol{\gamma})} \circ \left\langle \boldsymbol{u}^{s,(\boldsymbol{i})}, \boldsymbol{v}^{s,(\boldsymbol{j})} \right\rangle \right]_{\boldsymbol{ij}}$;

Both the straight forward and the inexact method can be used to obtain the coefficients of the expansions of the Craig Bampton internal modes and corresponding eigenvalues, achieving discretization of the stochastic space for uncertain Craig Bampton internal modes.

6.2.3 Gram-Schmidt

Both in the straight forward and the inexact method, the coefficients of the modes are orthonormalized by Gram-Schmidt. Let S be an independent set of k vectors in the subspace $\boldsymbol{R}^n$:

$$\boldsymbol{S} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_k\} \tag{6.51}$$

where k must be larger or equal to the number of dimensions of the subset n:

$$k \leq n \tag{6.52}$$

The Gram-Schmidt procedure can be used to generate an orthogonal set $\boldsymbol{S}'$ for the original set $\boldsymbol{S}$ which both span the same dimensional subspace:

$$\boldsymbol{S}' = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_k\} \tag{6.53}$$

by use of the projection:

$$\text{proj}_{\boldsymbol{u}}(\boldsymbol{w}) = \frac{\langle \boldsymbol{w}, \boldsymbol{u} \rangle}{\langle \boldsymbol{u}, \boldsymbol{u} \rangle} \boldsymbol{u} \tag{6.54}$$

where $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ denotes the inner product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. Two cases can be distinguished when applying the Gram-Schmidt projection: one for the first vector in the set and the other of all successive vectors. The procedure is the following. The first vector in the orthonormal set $\boldsymbol{S}'$ equals the first vector in the original set $\boldsymbol{S}$ normalized by its length:

$$\text{If } k = 1, \qquad \boldsymbol{u}_1 = \boldsymbol{w}_1, \qquad \boldsymbol{u}_{1,n} = \frac{\boldsymbol{u}_1}{|\boldsymbol{u}_1|} \tag{6.55}$$

where $\boldsymbol{u}_{1,n}$ is the normalized vector. All successive vectors in the orthonormal set $\boldsymbol{S}'$ are found by application of the Gram-Schmidt projection:

$$\text{If } k \neq 1, \qquad \boldsymbol{u}_k = \boldsymbol{v}_k - \sum_{l=1}^{k-1} \text{proj}_{\boldsymbol{u_l}}(\boldsymbol{w_k}), \qquad \boldsymbol{u}_{k,n} = \frac{\boldsymbol{u}_k}{|\boldsymbol{u}_k|} \tag{6.56}$$

where the set $\boldsymbol{u}_k$ is orthogonal and the set $\boldsymbol{u}_{k,n}$ is orthonormal.

If the Gram-Schmidt projection is implemented numerically, a loss of orthogonality can occur due to roundoff errors. By implementing a small alteration in the procedure, the loss of orthogonality can be minimized, numerically stabilizing the procedure. Instead of applying the summation in (6.56), the calculations are done step by step. The vector $\boldsymbol{a}_{k,b}$ will be used to denote the intermediate results, where the first subscript denotes which vector of the set is considered and the second subscript denotes the number of the intermediate result. The following is an example how to implement the modified Gram-Schmidt method for the first four vectors in the set:

$s = 1$	$s = 2$	$s = 3$	$s = 4$
$\boldsymbol{u}_1 = \boldsymbol{v}_1$	$\boldsymbol{a}_{2,1} = \boldsymbol{v}_2 - \text{proj}_{\boldsymbol{u}_1}(\boldsymbol{v}_2)$	$\boldsymbol{a}_{3,1} = \boldsymbol{v}_3 - \text{proj}_{\boldsymbol{u}_1}(\boldsymbol{v}_3)$	$\boldsymbol{a}_{4,1} = \boldsymbol{v}_4 - \text{proj}_{\boldsymbol{u}_1}(\boldsymbol{v}_4)$
	$\boldsymbol{u}_2 = \boldsymbol{a}_{2,1}$	$\boldsymbol{a}_{3,2} = \boldsymbol{a}_{3,1} - \text{proj}_{\boldsymbol{u}_2}(\boldsymbol{a}_{3,1})$	$\boldsymbol{a}_{4,2} = \boldsymbol{a}_{4,1} - \text{proj}_{\boldsymbol{u}_2}(\boldsymbol{a}_{4,1})$
		$\boldsymbol{u}_3 = \boldsymbol{a}_{3,2}$	$\boldsymbol{a}_{4,3} = \boldsymbol{a}_{4,2} - \text{proj}_{\boldsymbol{u}_3}(\boldsymbol{a}_{4,2})$
			$\boldsymbol{u}_4 = \boldsymbol{a}_{4,3}$
$\boldsymbol{u}_{1,n} = \frac{\boldsymbol{u}_1}{\lvert\boldsymbol{u}_1\rvert}$	$\boldsymbol{u}_{2,n} = \frac{\boldsymbol{u}_2}{\lvert\boldsymbol{u}_2\rvert}$	$\boldsymbol{u}_{3,n} = \frac{\boldsymbol{u}_3}{\lvert\boldsymbol{u}_3\rvert}$	$\boldsymbol{u}_{4,n} = \frac{\boldsymbol{u}_4}{\lvert\boldsymbol{u}_4\rvert}$

The modified Gram-Schmidt method is implemented to orthonormalize the coefficients of the expansions of the internal modes found using the straight forward or the inexact method.

7 Results

The reduction methods discussed and described in the previous chapters will be applied on the model of a flexible, three-dimensional slider crank to investigate the performance of the reduction methods on a flexible multibody dynamic benchmark problem.

7.1 Model description of the slider crank

The slider crank is a system consisting of two bars. One bar is connected to the real world by a hinge at point O such that the bar can rotate about O, but cannot translate. The other bar is connected at point B to a hinge on a cart. The cart can translate along the x-axis such that the movement of the second bar is restricted in the translation in y and z direction. The free ends of the bars are connected at point A. A schematic illustration of the slider crank is given in Figure 7.1. Both bars have a floating frame located in their center of mass (points 1 and 2). The x-direction of the floating frames is directed along the length of the bars. The global frame is referred to as O, but is located a length d along the y-axis away from the hinge at point O. The system is driven by a constant torque T applied at the hinge O and directed around the y-axis. The constant driving torque can only be applied in simulations over a relatively small period of time, because a constant driving torque results in continuously growing accelerations. For simulations over longer periods of time a harmonic torque should be applied instead.

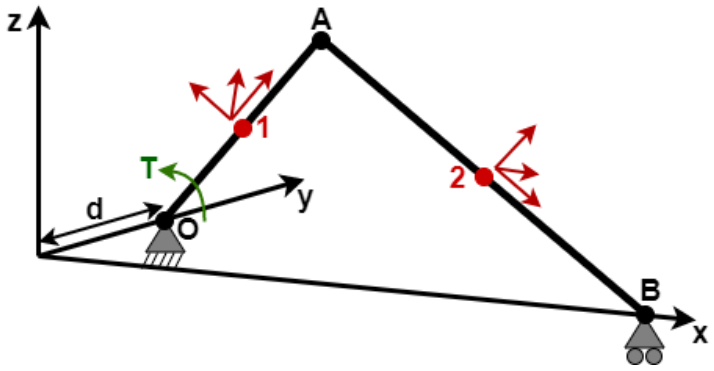


Figure 7.1: A schematic illustration of the slider crank.

The geometric and material properties of the slider crank are shown in Table 7.1. The shear and bulk modulus are the uncertain input parameters when stochastic problems are considered. Both parameters are log-normally distributed according to transformations of normal distributions with the characteristics show in Table 7.1. In case the deterministic problem is considered, the shear modulus and bulk modulus are equal to the average of their respective normal distributions. The FE stiffness matrix and mass matrix are dependent on the shear and bulk modulus. The expression for the FE stiffness matrix and mass matrix of the slider crank are shown in Appendix E.1, but are expressed in terms of the Young's modulus. The relation between the Young's modulus and the shear and bulk modulus is:

$$E(G, C) = \frac{9CG}{3C + G} \tag{7.1}$$

where E is the Young's modulus, G is the shear modulus and C is the bulk modulus. The relation between the Poisson's ratio and the shear and bulk modulus is:

$$\nu(G, C) = \frac{3C - 2G}{6C + 2G} \tag{7.2}$$

where ν is the Poisson's ratio. The Poisson's ratio is important because to model the slider crank as described in Chapter 3 the Poisson's ratio must be within the bounds:

$$0 \leq \nu \leq 0.5 \tag{7.3}$$

For the average values of the shear and bulk modulus the Poisson's ratio is within bounds. However, when the shear and bulk modulus are considered uncertain, the Poisson's ratio must be confirmed to be within bounds for every set of samples.

Description	**Quantity**	**Bar 1**	**Bar 2**	**Unit**
Length	L	0.15	0.3	m
Cross-sectional area	A	0.0001	0.0001	m^2
Density	ρ	7870	7870	kg/m^3
Shear modulus: average of the normal distribution	ν_G	76.9e9	76.9e9	Pa
Shear modulus: standard deviation of the normal distribution	σ_G	10e8	10e8	Pa
Bulk modulus: average of the normal distribution	ν_C	166.7e9	166.7e9	Pa
Bulk modulus: standard deviation of the normal distribution	σ_C	10e8	10e8	Pa
Number of elements	-	21	21	-
Distance between hinge O and the global frame	d	0.01		m
Driving torque	T	1		Nm

Table 7.1: Characteristics of the slider crank.

In the initial configuration both bars are positioned in the xy-plane with z-coordinates equal to zero while point A has positive x and y values. All velocities, accelerations and flexible deformations are zero in the initial configuration.

Because the slider crank has 12 floating frame coordinates and one independent coordinate, the rotation about the y-axis of the first bar, 11 constraints are required. The slider crank is defined by eight translation constraints: point O cannot translate in any direction, in point A the position of the ends of both bars must be the same and point B cannot translate in y and z direction. Furthermore, three rotation constraints are present. At point A the rotational axis is the same for both bars, which means the axis in the other two directions should be perpendicular to the rotational axis, as schematically illustrated in Figure 7.2. Finally, the rotation at point O is constraint such that the rotation is always perpendicular to the global z-axis. Table 7.2 gives the mathematical expression of the constraints.

Type	**Description**	**Expression**
3 translation constr. at O:	fixed position	$\boldsymbol{r}_1^{OO} + \boldsymbol{R}_1^O \boldsymbol{r}_O^{11}$
1 rotation constr. at O:	the rotation axis of the hinge at O is always perpendicular to the global z-axis	$(\boldsymbol{R}_1^O \boldsymbol{n}_y)^T \boldsymbol{n}_z$
3 translation constr. at A:	the ends of the bars are connected	$\boldsymbol{r}_1^{OO} + \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} - \boldsymbol{r}_2^{OO} - \boldsymbol{R}_2^O \boldsymbol{r}_A^{22}$
2 rotation constr. at A:	the rotation axis of the hinge in A is the local y-axis of both bars and always perpendicular to the other axes of the bars	$(\boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\boldsymbol{R}_2^O \boldsymbol{n}_y)$ $(\boldsymbol{R}_1^O \boldsymbol{n}_x)^T (\boldsymbol{R}_2^O \boldsymbol{n}_y)$
2 translation constr. at B:	cart can only slide along the x-axis	$(\boldsymbol{r}_2^{OO} + \boldsymbol{R}_2^O \boldsymbol{r}_O^{22})_{\text{row 2 and 3}}$

Table 7.2: Overview of the constraints.

Because the slider crank is three-dimensional, the rotation matrix must be updated when integrating the constrained equation of motion over time as explained in section 3.3.3. Therefore, expressions of the derivatives of the constraints which contain only the complete rotation matrices must be found. The derivatives can be obtained by performing the derivation by hand. Because time is not an explicit variable in the constraint equations, all derivatives of the constraints with respect to time equal zero. However, deriving the Jacobian and the derivative of the Jacobian to the generalized coordinates is tedious. The derivation of the Jacobian and its derivative can be found in Appendix E.2.

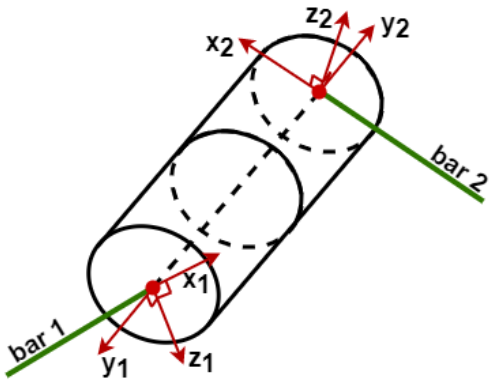


Figure 7.2: Illustration of the rotation constraints at the hinge.

7.2 Model reduction

Three techniques to achieve reduction in the spatial discretization have been discuss in chapter 4: free boundary modes, Craig Bampton and POD. Of the two physics-based reduction methods Craig Bampton is attractive over free boundary modes in case of flexible multibody dynamic problems defined using the floating frame definition, because of the use of the boundary nodes in both the floating frame definition and Craig Bampton [2]. Therefore, the performance of Craig Bampton reduction on the slider crank will be compared to the performance of the non-physics-based POD method, to investigate whether POD can achieve a similar or further reduction compared to Craig Bampton while maintaining an error of similar order between the reduced and unreduced simulations.

7.2.1 Craig Bampton

To compare the Craig Bampton reduction to POD, first a proper simulation of the slider crank using Craig Bampton must be achieved. To achieve a good simulation, one must consider whether the boundary modes provide enough detail to simulate the slider crank or whether internal modes should be included additionally and which of the three integration schemes proposed in chapter 3 performs best. The three integration methods proposed are application of forward and explicit Euler to integrate the constrained equations of motion directly (henceforth referred to as forward Euler), transforming the constrained equation of motion to an ODE and integrating the ODE by Runge-Kutta (henceforth referred to as Runge-Kutta) and integrating the embedded formulation of the constained equations of motion by Runge-Kutta. The latter has not been implemented due to time constraints.

To investigate whether to include the internal modes and which integration scheme to use, simulations of the slider crank will be done for the four different combinations of the settings. The simulations are done over a time period of 0.25 seconds. One can investigate the difference in results by comparing the trajectories of the floating frames and points O, A and B. The 15 coordinates will be referred to as $\boldsymbol{X}$. To determine the influence of including the internal modes, the error between the trajectory when using only the boundary modes and when adding the first twelve internal modes to the set of flexible modes shape is defined:

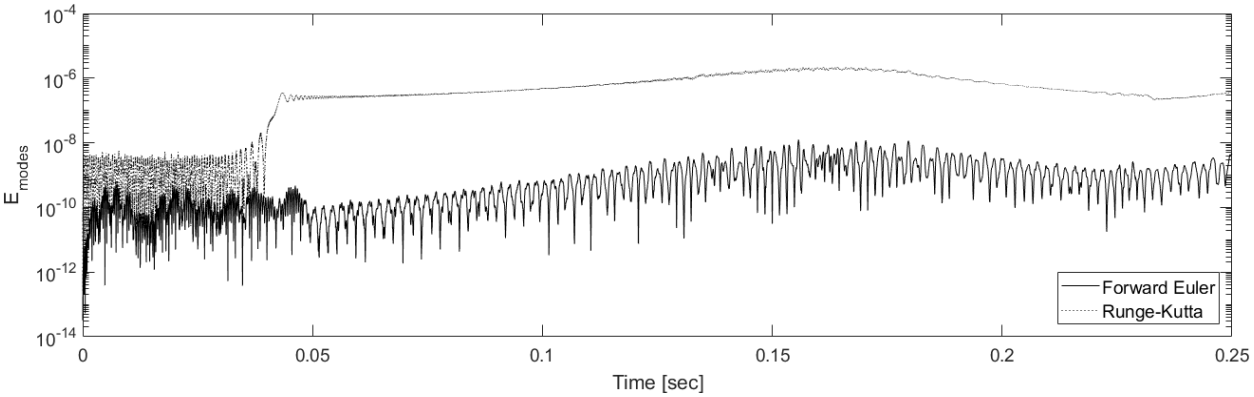
$$E_{modes}(t) = \frac{\sum \left(\boldsymbol{X}_b(t) - \boldsymbol{X}_{b+i}(t) \right)^2}{\sum \left(\boldsymbol{X}_b(t) \right)^2} \tag{7.4}$$

where E_{modes} is the error, $\boldsymbol{X}_b$ is the trajectory when only boundary modes are included and $\boldsymbol{X}_{b+i}$ is the trajectory in case the first twelve internal modes are included as well. The comparison is done using both integration schemes and the result is shown in Figure 7.3a. The integration schemes are compared by

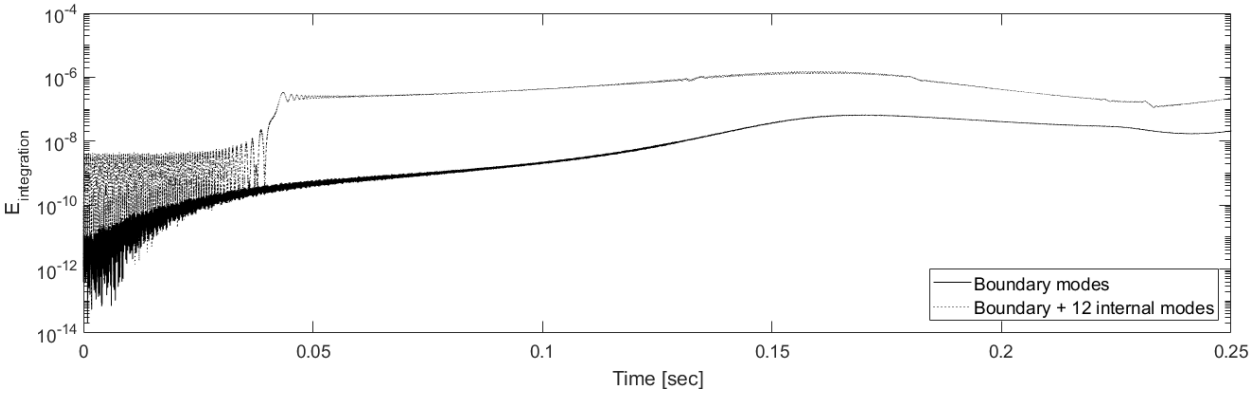
examining the difference between the trajectories as well:

$$E_{integration}(t) = \frac{\sum \left(\boldsymbol{X}_{FE}(t) - \boldsymbol{X}_{RK}(t) \right)^2}{\sum \left(\boldsymbol{X}_{FE}(t) \right)^2} \tag{7.5}$$

where $E_{integration}$ is the error, $\boldsymbol{X}_{FE}$ is the trajectory when integrating by forward Euler and $\boldsymbol{X}_{RK}$ is the trajectory when integrating by Runge-Kutta. A time step of 1e-6 seconds is applied in case of integration by forward Euler. The comparison is done once when using only the boundary modes and once when using both the boundary modes and the first twelve internal modes. The result is shown in Figure 7.3b.



(a) Error in trajectory $\boldsymbol{X}$ between using boundary modes and using 12 additional internal modes (7.4), integrated as listed in the legend.



(b) Error in trajectory $\boldsymbol{X}$ between integration by forward Euler and by Runge-Kutta (7.5), while using modes as listed in the legend.

Figure 7.3: The error between the resulting trajectory for the slider crank when using only the boundary modes or adding 12 internal modes and integrating by either forward Euler or Runge-Kutta.

As shown in Figure 7.3 the difference in trajectory between the different methods is small. Including or excluding the internal modes causes little change in the trajectory of the slider crank as shown in Figure 7.3a, which means using solely the Craig Bampton boundary modes provides sufficient detail to achieve a good simulation of the slider crank. Furthermore, adding the internal modes significantly extends the computation time, which makes the use of internal modes unattractive. The resulting trajectories when integrating the constrained equation of motion by forward Euler or Runge-Kutta is small as well, as can be seen in Figure 7.3b. Because of the small difference, one can conclude that no significant drift problems are encountered in the simulation of the slider crank due to the transformation of the DAE to an ODE. Because Runge-Kutta can be implemented with an adaptive time step, the computation time when applying Runge-Kutta is smaller compared to integration by forward Euler. Therefore, Runge-Kutta is preferred over forward Euler.

So, a proper simulation of the slider crank can be achieved when using only the boundary modes to represent the flexible behaviour and integrating the constrained equation of motion by Runge-Kutta. The resulting trajectory is shown in Figure 7.4, in which the thick continuous line shows the position of the slider crank after 0.25 seconds. The remaining lines represent the movement of the floating frames and points O, A and B over time.

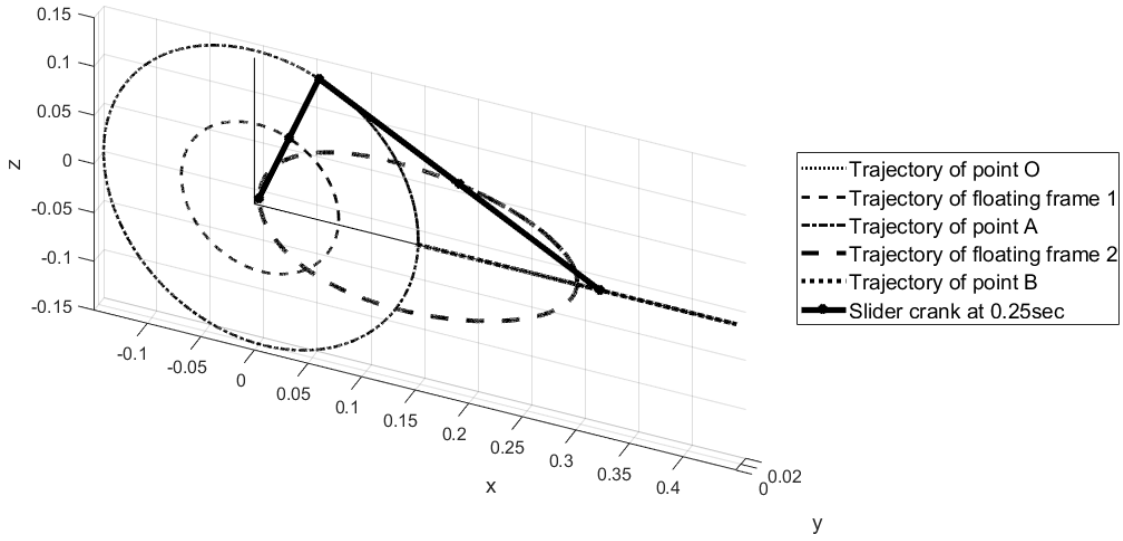


Figure 7.4: The trajectory of the slider crank in case only the Craig Bampton boundary modes are used and the system of equations is integrated by Runge-Kutta.

7.2.2 Quadratic velocity terms

The established simulation of the slider crank reduced by Craig Bampton provides an opportunity to investigate the assumption made in the derivation of the constrained equations of motion that the quadratic velocity terms can be neglected in the model of the slider crank. One can investigate the influence of the quadratic velocity terms on the trajectory of the slider crank by comparing the resulting trajectories when simulating the slider crank including and excluding the quadratic velocity terms:

$$E_{qvt}(t) = \frac{\sum \left(\boldsymbol{X}_{no\ qvt}(t) - \boldsymbol{X}_{qvt}(t) \right)^2}{\sum \left(\boldsymbol{X}_{no\ qvt}(t) \right)^2} \tag{7.6}$$

where E_{qvt} is the error, $\boldsymbol{X}_{no\ qvt}$ is the resulting trajectory in case the quadratic velocity terms are neglected and $\boldsymbol{X}_{qvt}$ is the resulting trajectory in case the quadratic velocity terms are included.

The size of the quadratic velocity terms depends on the velocities in the simulation. A rapidly rotating structure will have higher quadratic velocity terms. The size of the velocities in turn depends on input parameters such as the applied torque. If the torque is not harmonic, the velocities continually grow. Therefore, it is not only interesting to investigate the influence of the quadratic velocity terms in the model of the slider crank, but also to consider the effect of changing the applied torque. Figure 7.5 shows the error (7.6) when applied torques of 1, $\sin(4\pi t)$, 3 and $5\sin(4\pi t)$ Nm are considered.

The difference between the trajectories with and without quadratic velocities terms for each applied torque is small at first because the initial velocities of the slider crank initially are zero, resulting in small quadratic velocity terms. As time passes the velocity terms grow, resulting in growing quadratic terms as well. The growth rate is most rapid in case of a constant applied torque, which is as expected since the harmonic torques have a sin shape. Furthermore, a larger applied torque results in a larger error, because higher velocities are achieved sooner when a larger torque is applied. However, the error for all four torques is still small after 0.25 seconds. For the slider crank driven by the constant torque of 1 Nm, the error is in

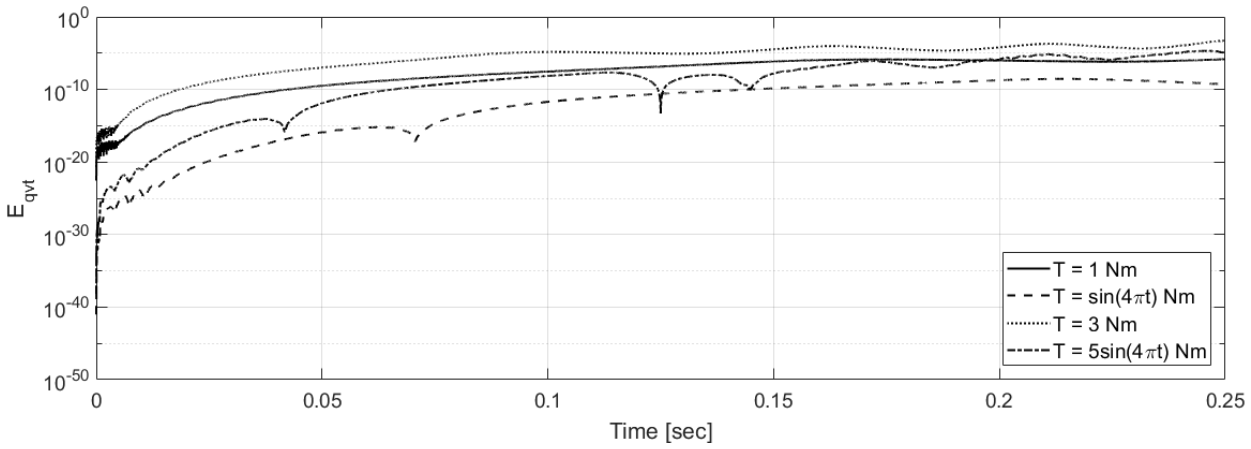


Figure 7.5: The error (7.6) in the trajectory $\boldsymbol{X}$ between simulating the slider crank including and excluding the quadratic velocity terms.

the range 1e-6 during the last 0.1 seconds of the simulation, which is small enough to conclude the quadratic velocity terms can be neglected in the simulation of the slider crank as described in section 7.1.

7.2.3 Proper orthogonal decomposition

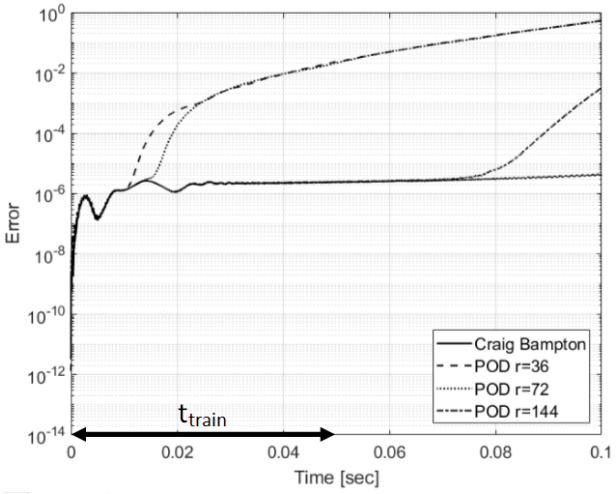
Alternatively to Craig Bampton, the model of the slider crank can be reduced using the non-physics-based method POD. In section 4.3, four levels to implement POD were suggested: application on system level and application on component level where either the two bars are defined as the two components, the flexible behaviour of the bars are defined as the two components while the rigid motion is not reduced or the flexible and rigid motion of the bars are defined as the four components. Furthermore, three parameters must be tuned in the application of POD: the number of samples m in the snapshot matrix, the time period over which the samples are collected t_{train} and the truncation order of the basis r. The question which arises is whether, by finding the best performing application level and tuning the parameters, one can achieve a reduction of the model of the slider crank by POD which surpasses the reduction by Craig Bampton, while the error between the unreduced and the POD and Craig Bampton reduced simulations is of the same order. The error between the unreduced and reduced simulations is defined as the difference between the trajectories of the floating frames and points O, A and B:

$$E(t) = \frac{\sum \left(\boldsymbol{X}_{UR}(t) - \boldsymbol{X}_{POD}(t) \right)^2}{\sum \left(\boldsymbol{X}_{UR}(t) \right)^2} \tag{7.7}$$

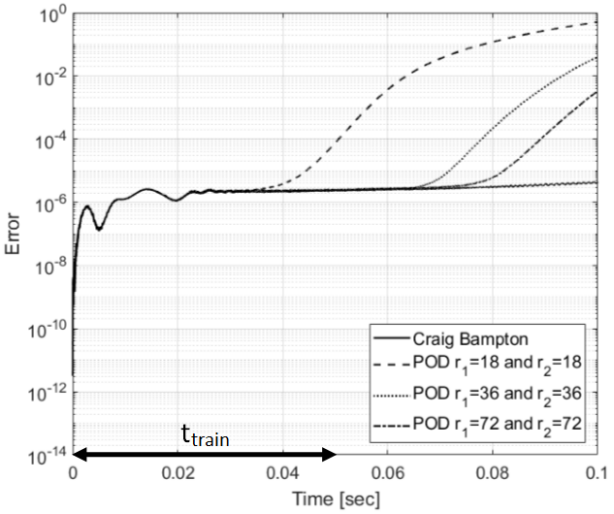
where E is the error, $\boldsymbol{X}_{UR}$ is the trajectory of the unreduced simulation and $\boldsymbol{X}_R$ is the trajectory of the reduced simulation, either by POD or Craig Bampton.

The application level and truncation order. First, the quality of the results of applying POD on the four different levels while varying the truncation order will be examined. To compare the results, the reduced models of the slider crank are integrated using Runge-Kutta over 0.1 seconds. For each reduction, a snapshot matrix containing 50 samples collected over a training of 0.05 seconds is constructed. The resulting errors in trajectory (7.7) between the POD reduced models and the unreduced model and the error between the Craig Bampton reduced and unreduced model are shown in Figure 7.6. When considering the graphs, one should note that the unreduced POD basis has 252 columns while the number of modes used in the application of Craig Bampton is 36 (6 rigid and 12 flexible modes per bar).

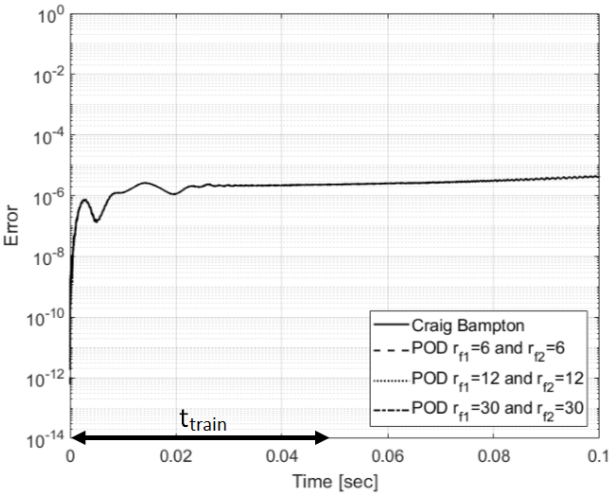
The POD reduction on system level (Figure 7.6a) does not perform well. If 144 of the 252 POD modes are used, the error is similar to the Craig Bampton error for the first 0.07 second, after which the error increases rapidly. When one reduces the bases further to include 72 or 36 modes, the error grows unacceptably



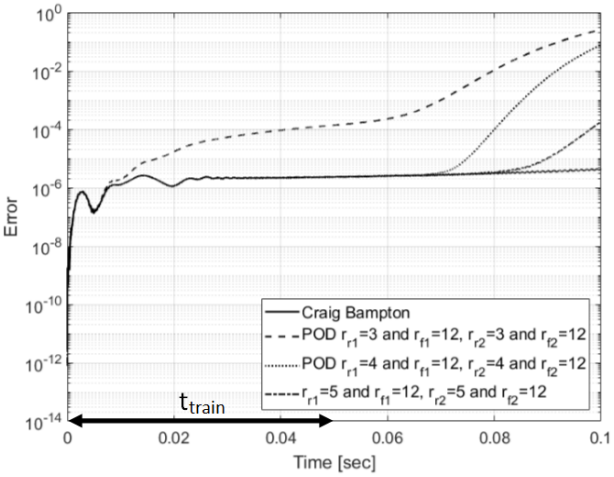
(a) System level, where r is the reduction order.



(b) Component level: 2 bars, where r_1 and r_2 are the reduction orders of the first and second bar respectively.



(c) Component level: flexible behaviour of the 2 bars, where r_{f1} and r_{f2} are the reduction order of the flexible behaviour of the first and second bar respectively.



(d) Component level: rigid and flexible behaviour of the 2 bars, where r_{r1} and r_{r2} are the reduction order of the rigid behaviour of the first and second bar respectively and r_{f1} and r_{f2} are the reduction order of the flexible behaviour of the first and second bar respectively.

Figure 7.6: The error in trajectory (7.7) between the unreduced model of the slider crank and the model reduced by Craig Bampton or POD when POD is applied on different levels with varying truncation orders.

large within the training time. By defining the two bars of the slider crank as components (Figure 7.6b) the result is improved. The moment the error of POD and the error of Craig Bampton start to deviate is delayed. However, the error still grows large for all three truncation orders, leading to the conclusion that application of POD on the bars separately does not yield good results either. Promising results are obtained when applying POD on the flexible behaviour of the two bars separately without reducing the rigid movement (Figure 7.6c). The error between the unreduced model and the reduced models by POD and Craig Bampton cannot be distinguished, even when reducing the number of singular vectors below the number of Craig Bampton modes. However, if the rigid motion is also reduced (Figure 7.6d) the error of the POD simulations again grows large. One can conclude that the flexible behaviour of the slider crank can be reduced significantly using POD, but the rigid motion cannot. Why the rigid motion cannot be reduced is likely due to the non-linearities in the rigid motion of the slider crank, because POD is a linear reduction

technique which therefore performs best when applied to linear problems [18]. Applying POD to reduce the flexible behaviour only is similar to the application of Craig Bampton, since Craig Bampton is also used to reduce the flexible generalized coordinates, not the rigid motion.

To investigate the quality of the POD reduction applied on the flexible behaviour only, the rigid motion should be excluded in the calculation of the error (7.7), because the rigid motion is several orders larger compared to the flexible motion. By subtracting the rigid motion from the trajectories before application of (7.7), the difference in flexible behaviour between the unreduced and reduced models can be examined. Table 7.3 shows the average value of the errors after every 1e-5 seconds when the model is integrated over 0.1 seconds. The lowest truncation order for which the POD reduced simulation has a lower error compared to the Craig Bampton reduced simulation is three for both components. From the singular values of the two snapshot matrices shown in Figure 7.8 it is clear that the first two POD modes of both components contain the dominant flexible behaviour, explaining the rapid decrease of the error in flexible behaviour in case the first, first two or first three POD modes are used. If more singular vectors are included in the basis, the error still decreases, but far less rapidly. As expected, a POD reduction applied only on the flexible modes with a truncation order of three results in a rigid motion with a similar error compared to Craig Bampton as can be seen in Figure 7.7. So, the smallest POD basis with an error in rigid and flexible motion which is of the same order as the error between the unreduced and Craig Bampton reduced model contains three POD modes for the flexible behaviour of each bar. The POD basis contains thus a total of 18 modes which is less compared to the 36 modes used in the application of Craig Bampton.

	Mean error
CB	0.790
POD $r_{f1} = r_{f2} = 1$	1.538
POD $r_{f1} = r_{f2} = 2$	0.864
POD $r_{f1} = r_{f2} = 3$	0.678
POD $r_{f1} = r_{f2} = 6$	0.741
POD $r_{f1} = r_{f2} = 12$	0.715
POD $r_{f1} = r_{f2} = 30$	0.141

Table 7.3: Mean error of the trajectory of only the flexible coordinates, where r_{f1} and r_{f2} are the reduction order of the flexible behaviour of the first and second bar respectively.

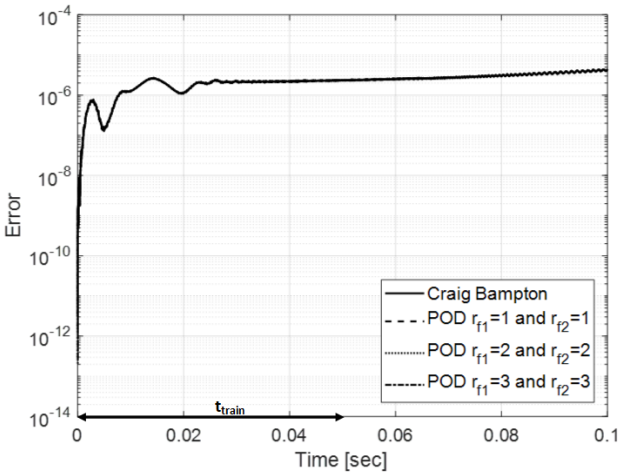
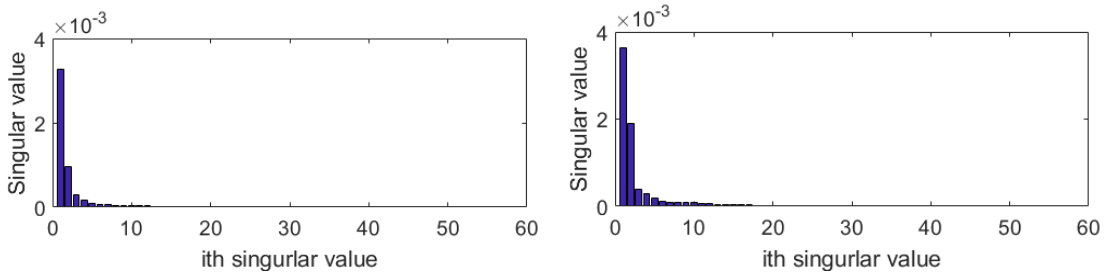


Figure 7.7: Applying POD in time on the slider crank on the flexible behaviour of the 2 bars separately, where r_{f1} and r_{f2} are the reduction order of the flexible behaviour of the first and second bar respectively.



(a) Singular values of bar 1.

(b) Singular values of bar 2.

Figure 7.8: The singular values of the snapshot matrices of the flexible behaviour of both bars.

The number of samples and training time. The result of the application of POD on the flexible behaviour of both bars separately with a truncation order of three might be improved further by tuning the number of samples m in the snapshot matrix and the training time t_{train}. The mean of the error in the complete trajectory and in the flexible part of the trajectory for POD reductions with different numbers of samples and training times are shown in Table 7.4. As reference, the mean of the resulting error between the unreduced simulation and the Craig Bampton simulation for the complete trajectory is 2.393e-6 and for the flexible part is 0.790.

	Mean error in trajectory				**Mean error in flexible trajectory**			
m / t_{train} [s]	**50**	**25**	**12**	**6**	**50**	**25**	**12**	**6**
0.050	2.417e-6	2.408e-6	2.426e-6	2.388e-6	0.6778	0.8986	1.117	1.673
0.025	2.414e-6	2.421e-6	2.416e-6	2.411e-6	0.5727	0.8920	1.048	1.270
0.012	2.409e-6	2.410e-6	2.402e-6	2.399e-6	0.6439	0.8025	1.511	1.768
0.006	2.418e-6	2.405e-6	2.392e-6	2.409e-6	0.8537	0.9069	1.108	1.427

Table 7.4: The mean error over time between the unreduced and POD reduced simulations where POD is applied on the flexible behaviour of the two bars separately, the system is integrated over 0.1s and the number of samples (m) in the snapshot matrix and the training time (t_{train}) over which the samples are obtained are as defined in the table.

Regardless of the number of samples and the training time, the mean error in the complete trajectory is always about 2.4e-6, which is in the same range as the mean error when using Craig Bampton. The fact that the errors of the complete trajectory when using POD and Craig Bampton are similar is expected, since the rigid motion is not reduced in either method. When considering the flexible behaviour only, the training time does not have a big influence on the error either. However, it is expected that if the simulation is integrated over a longer period of time, the basis might not be representative of the movement any longer resulting in a growing error [17]. The number of samples in the snapshot matrix does have a clear influence: a low number of samples results in a higher error. The basis contains the dominant behaviour of the slider crank, which is determined using the snapshot matrix. If too few samples are used in the snapshot matrix, important behaviour might be be captured in the bases. In case 50 samples are used, the error of the POD reduction is lower compared to Craig Bampton. When including 25, 12 or 6 samples, the error is higher.

So, the furthest POD reduction of the slider crank model which has a similar error in trajectory of the rigid and flexible behaviour compared to Craig Bampton is obtained when applying POD on the flexible behaviour of the two bars separately with a snapshot matrix of 50 sample collected over a training time of 0.012s and reducing the bases to 3 singular vectors each. The complete POD basis contains 18 modes: six for the rigid behaviour of both bars and three for the flexible behaviour of both bars, which surpasses the reduction achieved using Craig Bampton.

7.3 Model uncertainties

In section 5, PCE is proposed as method to discretize the stochastic space of the stochastic constrained equation of motion. The performance of PCE is investigated using the model of the slider crank. PC expansions are made which approximate the x-coordinate of point A on the slider crank after 0.1 seconds in case only the shear modulus is uncertain and in case both the shear and bulk modulus are uncertain. Leave-one-out cross validation is used to evaluate the performance of the expansions (see section 5.1.2), with the aim to obtain PC expansions which approximate the model of the slider crank closely but reduce the evaluation time such that the application of Monte Carlo on the expansions is feasible.

7.3.1 Uncertain shear modulus

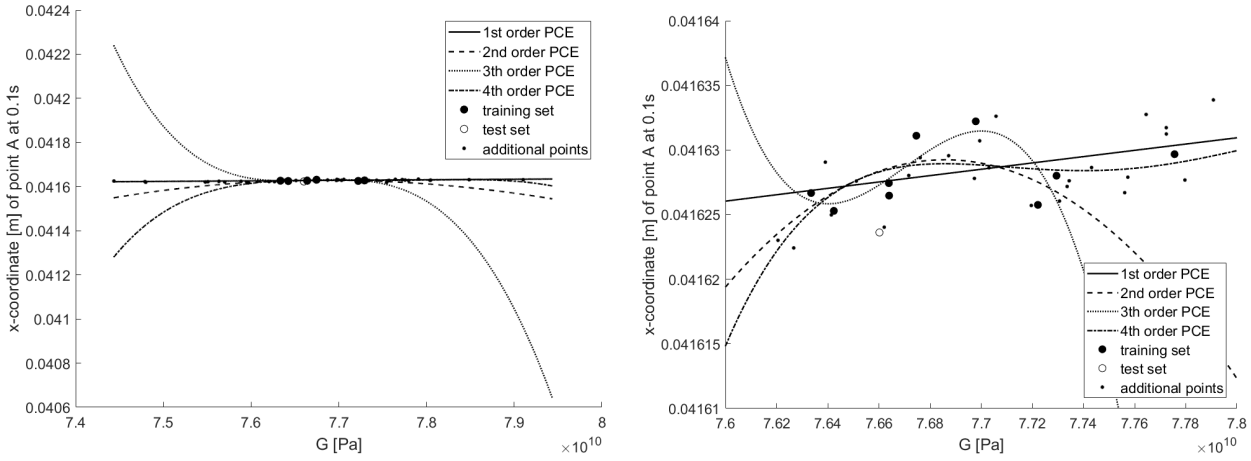
The case of an uncertain shear modulus is considered first. The result of the leave-one-out cross validation for expansions of different orders is shown in Table 7.5. The expansions with a truncation order up to three have a similar and very small error. The error grows for expansions with higher truncation orders, indicating the polynomials in the expansions are overfitted on the data in case the truncation order is higher than 3.

For truncation orders from one to three, the cross validation errors are of the order 1e-9, which means the expansions approximate the original model well. Because the errors are similar, it is difficult to determine which truncation order is best suited to approximate the stochastic model of the slider crank using the leave-one-out cross validation. Instead, the expansions will be visually examined.

Truncation order p	**Error** $E_{average}$
1	8.324e-9
2	7.304e-9
3	8.557e-9
4	8.304e-6
5	8.763e-7
6	6.161e-7

Table 7.5: Leave-one-out cross validation error for different truncation orders when varying the shear modulus.

Figure 7.9 shows the expansions of order 1, 2, 3 and 4, as well as the training data set, test set and additional random samples. The complete training set is used for the fourth order expansion, while only a section of the training set is used for the lower order expansions. Figure 7.9a shows the expansions for a large interval of the shear modulus. The training and test points are concentrated in the middle of the x-axis, which is expected since the distribution of G is log-normal. Around the training set the expansions are very similar. However, for points further away from the training set, the metamodels give very different results. The first and second order expansions represent the additional point relatively well, but the third and fourth order expansion do not, confirming that expansion with too high an order give poor results outside the training data set. Figure 7.9b zooms in on the centre of Figure 7.9a and clearly shows that the third and fourth order expansions are indeed overfitted on the training data set.



(a) The expansions.

(b) A zoom in of the left plot.

Figure 7.9: The PC expansions for different orders along with the training and test sets and additional randomly sampled points when varying the shear modulus.

From figure 7.9 and the leave-one-out cross validation, one can concluded that the best expansion order to use for the metamodel of the slider crank when only the shear modulus is considered uncertain is first order. Evaluation of the first order expansion takes about 0.0004 seconds, while, using the same computer, evaluation of the constrained equation of motion takes about 120 seconds. The reduction is significant. In conclusion, the first order PC expansion of the x-coordinate of point A of the slider crank after 0.1 seconds in case of an uncertain shear modulus approximates the stochastic constrained equation of motion closely and reduces the evaluation time such that application of Monte Carlo for a million samples on the first order expansion is feasible.

7.3.2 Uncertain shear and bulk modulus

The expansion for the slider crank when both the shear and bulk modulus are considered uncertain is examined now. The results of the leave-one-out cross validation for expansions of different orders as given in Table 7.6. The result of the leave-one-out cross validation shows that the expansion of order three performs significantly better compared to higher and lower expansion orders. For the first order approximation, the size of the error is partly due to the matrix $\boldsymbol{A}^T\boldsymbol{A}$, which is used to find the coefficients, being close to singular. For the other truncation orders, the difference in cross validation errors is due to overfitting when the order is too high or, when the order is too low, the expansion not being able to follow the trend of the output. The expansions can again be plotted to confirm that using expansions of higher and lower order than three are over- and underfitted on the training data.

Truncation order p	**Error** E_{LOO}
1	2.457e2
2	2.240e-6
3	5.652e-8
4	2.599e-7
5	5.966e-7
6	7.977e-6

Table 7.6: Leave-one-out cross validation error for different truncation orders when varying the shear and bulk modulus.

Figure 7.10a shows the second and third order expansions. The expansions cover the same range of x-coordinates but are shaped differently. Even though it is difficult to see without being able to rotate the figure, the third order approximation follows the shape of the data points more precisely because of the additional curves in the plane. Figure 7.10b shows the third and fourth order expansions. The fourth order approximation rises steeply in x-values around the minimum and maximum of the C and G axis, indicating, similar to the results seen in Figure 7.9, that the fourth order expansion is overfitted on the training data as was expected.

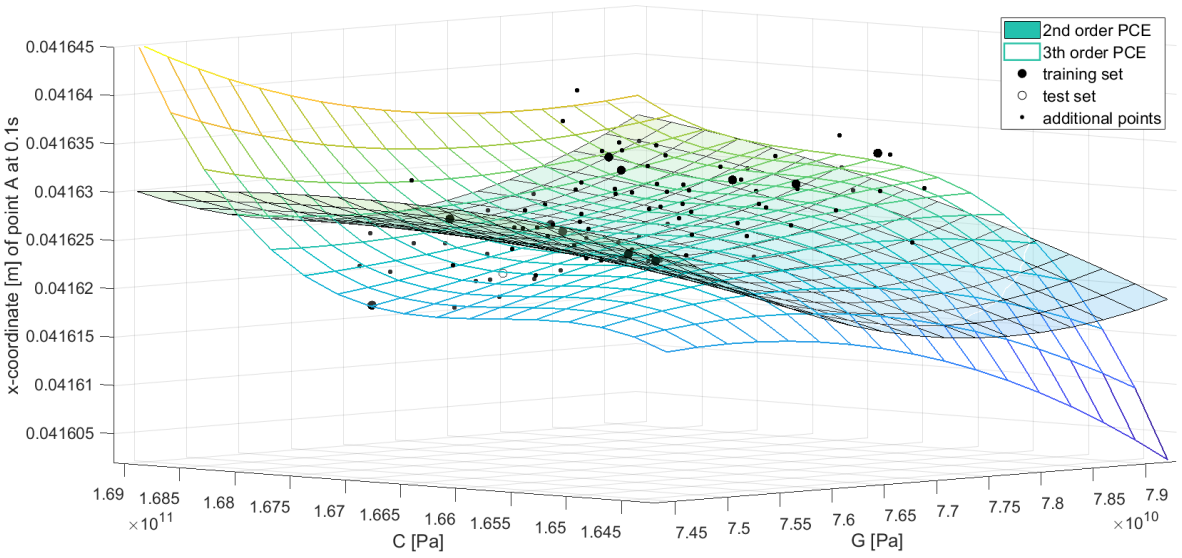
From the leave-one-out cross validation and the visual comparison of the expansions, it can be concluded that the third order expansion approximates the model of the slider crank closest when both the shear and bulk modulus are uncertain. The small cross validation error indicates that the third order PC expansion approximates the original model well. The evaluation time of the expansion is about 0.0005 seconds, which means that application of Monte Carlo for a million samples on the expansion is feasible.

7.4 POD based Craig Bampton

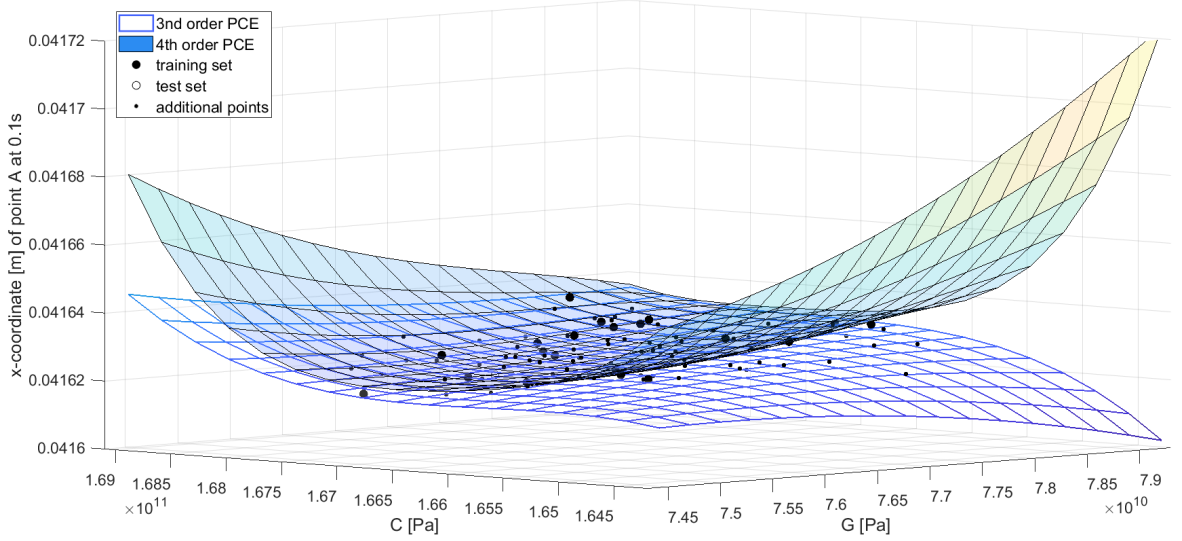
The FE stiffness matrix is dependent on the shear modulus. Both the boundary and internal Craig Bampton modes are dependent on the stiffness matrix and are thus uncertain in case of an uncertain shear modulus. In section 6.1, POD is purposed as a method to discretize the stochastic space. The POD reductions will be applied on the left bar of the slider crank in case of an uncertain shear modulus and the results are compared to the original model to investigate the performance of POD and examine the relation between POD modes and Craig Bampton modes.

In the calculation of the POD basis two parameters must be tuned: the number of samples in the snapshot matrix m and the reduction order r. To validate the results of POD and to determine which combination of the parameters yields the best results, leave-one-out cross validation is applied. A set of $m+1$ samples of the shear modulus is obtained through random sampling and is sorted. Of the set, m samples are used to form the snapshot matrix and called the training set. The remaining point is used as test set to determine the error between the boundary and internal modes obtained through POD and the modes obtained by solving the original problem for the test sample. Each sample in the set is used as test set once and the average of the resulting error is used to evaluate the performance of POD for different choices for m and r.

The error between the Craig Bampton boundary modes of the original problem and the boundary modes



(a) The expansions of 2nd and 3th order.



(b) The expansions of 3th and 4th order.

Figure 7.10: The PC expansions for different orders along with the training and test sets and additional randomly sampled points when varying the shear and bulk modulus.

obtained through POD is defined as

$$E^s_{CB} = \frac{\sum \left(\mathbf{\Phi}^{s,POD}_{CB} - \mathbf{\Phi}^{s,MC}_{CB} \right)^2}{\sum \left(\mathbf{\Phi}^{s,MC}_{CB} \right)^2} \qquad \forall s = 1, 2, 3, ..., n_{CB} \tag{7.8}$$

where E^s_{CB} is the error between the s'th boundary mode obtained by solving the deterministic problem for a sample of the shear modulus $\mathbf{\Phi}^{s,MC}_{CB}$ (Monte Carlo) and the boundary mode obtained through POD $\mathbf{\Phi}^{s,POD}_{CB}$. To compare the internal modes obtained through Monte Carlo and POD, one must take into account that the shape of the modes should be the same, but the direction and magnitude of the deformation can differ because every linear combination of eigenvectors with the same eigenvalue is another eigenvector in the same eigenspace. The internal modes can be compared by finding a linear combination of the modes by POD

which approximates the mode by Monte Carlo:

$$\begin{bmatrix} \boldsymbol{\Phi}_{eig}^{p,POD} & \boldsymbol{\Phi}_{eig}^{q,POD} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \approx \boldsymbol{\Phi}_{eig}^{s,MC} \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{7.9}$$

where $\boldsymbol{\Phi}_{eig}^{p,POD}$ and $\boldsymbol{\Phi}_{eig}^{q,POD}$ are modes obtained through POD with the same eigenvalue as the s'th mode obtained through Monte Carlo $\boldsymbol{\Phi}_{eig}^{s,MC}$ which are all deformed on the same side of the bar and a and b are coefficients determined by least-squares. The error of the internal modes is defined as the error between the linear combination of POD modes and the Monte Carlo mode:

$$E_{eig}^{s} = \frac{\sum \left(\begin{bmatrix} \boldsymbol{\Phi}_{eig}^{p,POD} & \boldsymbol{\Phi}_{eig}^{q,POD} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} - \boldsymbol{\Phi}_{eig}^{s,MC} \right)^2}{\sum \left(\boldsymbol{\Phi}_{eig}^{s,MC} \right)^2} \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{7.10}$$

where E_{eig}^{s} is the error for the s'th internal mode. The error between the eigenvalues of the internal modes by POD and Monte Carlo is defined as:

$$E_{eig,\lambda}^{s} = \frac{\left((\boldsymbol{\lambda}_{eig}^{s,POD})^2 - (\boldsymbol{\lambda}_{eig}^{s,MC})^2 \right)^2}{\left((\boldsymbol{\lambda}_{eig}^{s,MC})^2 \right)^2} \qquad \forall s = 1, 2, 3, ..., n_{eig} \tag{7.11}$$

where $E_{eig,\lambda}^{s}$ is the error between the s'th eigenvalue obtained through POD ($\boldsymbol{\lambda}_{eig}^{s,POD}$) and the s'th eigenvalue obtained through Monte Carlo ($\boldsymbol{\lambda}_{eig}^{s,MC}$).

The validation method is described in mathematical terms in Algorithm 4.

7.4.1 Boundary modes

The results of the leave-one-out cross validation for the Craig Bampton boundary modes is shown in Table 7.7, in which the errors are the average errors of the cross validation of all 12 Craig Bampton modes.

		Error $\boldsymbol{\Phi}_{CB}$			
r		**10**	**5**	**2**	**1**
m	**100**	7.67e-35	1.44e-31	1.21e-30	6.40e-30
	50	9.01e-35	1.86e-31	1.42e-30	7.20e-30
	10	1.72e-33	3.60e-31	2.33e-30	1.00e-29
	5	6.74e-31	5.75e-31	3.42e-30	1.15e-29

Table 7.7: The results of the leave-one-out cross validation for the Craig Bampton modes with different truncations of the basis r and different numbers of samples in the snapshot matrix m, when the shear modulus is uncertain.

The difference between the Craig Bampton modes obtained by POD and by Monte Carlo is almost zero, as all errors shown in Table 7.7 are very small. The parameters m and r do seem to have an influence on the resulting error. A higher reduction order r leads to a smaller error because a larger number of singular vectors are included in the basis. However, including only one column is enough to obtain a good approximation. Regardless of which reduction order is used, the resulting solution for $\boldsymbol{a}_r^s$ is always very small (in the range 1e-15). The solution is small because the shear modulus is constant throughout the bar, meaning the effect of a small change in shear modulus has little effect on the Craig Bampton boundary modes. Because the shear modulus has little influence, the centralizing term $\bar{\boldsymbol{\Phi}}^s$ is enough to obtain a good approximation of the modes. The basis does still provide additional detail considering the small influence the shear modulus has, because the error does decrease for higher reduction order. Therefore, a larger r leads to a better result, even though the effect is small. The influence of the number of samples in the snapshot matrix on the error is less clear. From the results when r equals 10 or 1, it seems that a higher m leads to a smaller error. However,

Algorithm 4: Leave-one-out cross validation for the application of POD on the modes.

Input : m, r, $\mu_{G,ln}$, $\sigma_{G,ln}$, ϵ_t
Output: E^s_{mean}

Randomly sample a standard normal distribution: $\boldsymbol{\xi} = \text{randn}(m)$;
Transform the distribution to find samples for G: $\boldsymbol{G} = e^{\mu_{G,ln}+\sigma_{G,ln}\boldsymbol{\xi}}$;
for $i = 1 : m$ **do**
  Separate the samples in a train and test set: $\boldsymbol{G}_{tr} = \boldsymbol{G}([1 : \text{m} - 1, \text{m} + 1 : \text{end}])$, $G_{te} = \boldsymbol{G}(m)$;
  Determine the POD basisses: $\boldsymbol{U}^s\boldsymbol{\Sigma}^s(\boldsymbol{V}^s)^T = \left[\boldsymbol{\Phi}^s(\boldsymbol{G}_{tr})\right] - \bar{\boldsymbol{\Phi}}^s$;
  Reduce the basis: $\boldsymbol{U}^s_r = \boldsymbol{U}^s(:, 1 : r)$;
  if *Considering boundary modes* **then**
    Find the reduced variables for the test set: $\boldsymbol{a}^s_r = (\boldsymbol{U}^s_r)^T(-\boldsymbol{K}^{-1}_{ii}(G_{te})\boldsymbol{K}_{ib}(G_{te}) - \bar{\boldsymbol{\Phi}}^s)$;
    Determine the error: $E^s_{CB} = \dfrac{\sum\left(\boldsymbol{U}^s_r\boldsymbol{a}^s_r + \bar{\boldsymbol{\Phi}}^s - \boldsymbol{\Phi}^{s,MC}_{CB}\right)^2}{\sum\left(\boldsymbol{\Phi}^{s,MC}_{CB}\right)^2}$;
  else
    Find the reduced variables for the test set: $\left[(\hat{\boldsymbol{U}}^s_r)^T\boldsymbol{K}_{ii}(G_{te})\hat{\boldsymbol{U}}^s_r\right]\hat{\boldsymbol{a}}^s_r = (\lambda^s)^2\left[(\hat{\boldsymbol{U}}^s_r)^T\boldsymbol{M}_{ii}\hat{\boldsymbol{U}}^s_r\right]\hat{\boldsymbol{a}}^s_r$;
    Select the two resulting modes with the same eigenvalue ($\hat{\boldsymbol{a}}^s_{r,1}$ and $\hat{\boldsymbol{a}}^s_{r,2}$) and define the solution $\hat{\boldsymbol{a}}^s_r$ as the sum of said modes of which the first element equals one: $\hat{\boldsymbol{a}}^s = \frac{1}{\hat{\boldsymbol{a}}^s_{r,1}(1)+\hat{\boldsymbol{a}}^s_{r,2}(1)}(\hat{\boldsymbol{a}}^s_{r,1} + \hat{\boldsymbol{a}}^s_{r,2})$;
    Determine the best linear combination: $\min\left\{\begin{bmatrix}\hat{\boldsymbol{U}}^p_r\hat{\boldsymbol{a}}^p_r & \hat{\boldsymbol{U}}^q_r\hat{\boldsymbol{a}}^q_r\end{bmatrix}\begin{bmatrix}a\\b\end{bmatrix} - \boldsymbol{\Phi}^{s,MC}_{eig}\right\}$;
    Determine the error: $E^s = \dfrac{\sum\left(\begin{bmatrix}\hat{\boldsymbol{U}}^p_r\hat{\boldsymbol{a}}^p_r & \hat{\boldsymbol{U}}^q_r\hat{\boldsymbol{a}}^q_r\end{bmatrix}\begin{bmatrix}a\\b\end{bmatrix} - \boldsymbol{\Phi}^{s,MC}_{eig}\right)^2}{\sum\left(\boldsymbol{\Phi}^{s,MC}_{eig}\right)^2}$;
  end
end
Determine the average error: $E^s_{mean} = \frac{1}{m}\sum \boldsymbol{E}^s$;

the results for r equals 5 and 2 do not show the effect. However, every value for m shown in the table results in a small leave-one-out cross validation error.

Because the errors for the considered combinations of r and m are small, it can be concluded that the POD reduction for the Craig Bampton modes yields good results. So, the Craig Bampton modes are a linear combination of the POD modes, showing a relation between the two methods.

7.4.2 Internal modes.

The result of the leave-one-out cross validation of the constrained eigenmodes and values is shown in Table 7.8. The shown values are the average cross validation errors of the first eight eigenmodes and values of the bar.

Unlike the boundary modes the constrained eigenvalues are influenced by the shear modulus, explaining the larger errors. However, the errors for both the internal modes and corresponding eigenvalues are still relatively small, indicating POD provides a proper discretization of the stochastic space. For most combinations of m and r the cross validation error is in the same range, demonstrating that the parameters have little influence of the outcome. However, the error of the eigenmodes significantly increases in case the truncation order is equal or higher compared to the number of samples in the snapshot matrix. Whether the condition $r \leq m$ holds for all values for r and m should be investigated further. Because, as long as $r \leq m$ holds, the reduction order does not influence on the resulting error strongly, it is recommended to use the smallest reduction order r=1. Using r=1 does not only reduce the computation time because the matrices in the eigenvalue problem are as small as can be achieved by POD, but also because using r=1 removes the

		Error Φ_{eig}				Error λ^2_{eig}			
r		**10**	**5**	**2**	**1**	**10**	**5**	**2**	**1**
m	**100**	1.14e-23	3.70e-23	8.72e-24	8.90e-24	1.59e-24	2.91e-25	1.42e-25	1.56e-25
	50	9.27e-24	8.07e-24	1.75e-22	1.03e-23	1.88e-24	1.88e-25	1.71e-25	1.56e-25
	10	1.40e-14	5.17e-23	9.14e-24	1.03e-23	5.46e-24	8.72e-25	1.98e-25	2.28e-25
	5	2.47e-13	1.62e-13	1.18e-23	7.55e-24	1.20e-21	6.75e-25	2.06e-25	2.67e-25

Table 7.8: The results of the leave-one-out cross validation for the constrained eigenmodes and values with different truncations of the basis r and different numbers of samples in the snapshot matrix m, when the shear modulus is uncertain.

need of identifying the two equal eigenvalues since no other eigenvalues are found. It can be concluded that applying POD on the constrained eigenmodes yields good results, especially when using a higher number of snapshot compared to the value of the truncation order, meaning that the internal Craig Bampton modes are linear combinations of the POD modes.

7.5 Stochastic Galerkin on internal modes

In section 6.2, stochastic Galerkin is proposed as a method to discretize the stochastic space of the uncertain internal Craig Bampton modes. The performance of stochastic Galekin is examined using the model of the slider crank. PC expansions will be attained for the internal modes and corresponding eigenmodes of the first bar of the slider crank in case only the shear modulus is uncertain and in case both the shear and the bulk modulus are uncertain. The coefficients of the expansions will be determined using both the straight forward and the inexact method to be able to compare the performance of the methods.

The performance of stochastic Galerkin will be evaluated using three different validations which will be applied on the expansions of the eigenvalues only. If good results are obtained, the shape of the modes will be examined.

Validation 1: expansion orders. The first validation aims to determine which orders should be used for the PC expansions of the internal stiffness matrix $\boldsymbol{K}_{ii}$, the internal modes $\boldsymbol{u}^s$ and the corresponding eigenvalues λ^s. Leave-one-out cross validation is applied to compare the performance of expansions of different orders. A set of sample points of the shear and bulk modulus is randomly collected, of which all but one are used to find the coefficients of the stiffness matrix through regression on the training set since the shear and bulk modulus are not linearly related to the stiffness matrix. Using the obtained coefficients of the stiffness matrix, the PC coefficients of the eigenmodes and values are determined. The remaining point in the sample set is used to deine the error between the expansions of the eigenvalues and Monte Carlo:

$$E_{\lambda^s} = \left(\frac{\lambda^s_{MC} - \lambda^s_{SG}}{\lambda^s_{MC}} \right)^2 \tag{7.12}$$

where E_{λ^s} is the error between the s'th eigenvalue found using Monte Carlo (λ^s_{MC}) and the s'th eigenvalue found through stochastic Galerkin (λ^s_{SG}). Every point in the sample set is used for testing once. One can use the average error to compare the performance of expansions of different orders. From the results, the combination of orders of PC expansions which performs best results is selected and will be used in the remaining two validation methods.

Validation 2: variance and mean. The variance and mean of the expansion for the eigenvalues should be the same as the variance and mean of the solution of the original system. If the variance and mean are not at least comparable, the PC expansion does not approximate the internal eigenvalues of the first bar of the slider crank well. The comparison is done by applying Monte Carlo on the original system and on the PC expansion for 1e6 samples of the shear and bulk modulus. The resulting values are used to determine the probability distributions which are plotted in the same figure. The variance and mean of both distribution should be closely similar or equal, meaning the graphs should be (almost) perfectly aligned.

Validation 3: zeroth order coefficients. The zeroth order coefficients of the expansion should be equal to the solution of the original system in case the uncertain variables are equal to their respective means, because if all uncertain variables have the expected value, the only terms in the expansions which are not zero are the zeroth order terms. Therefore, the solution of the original system for the mean uncertain variables is compared to the zeroth order coefficients of the expansion by:

$$E_{0th,\lambda^s,SG} = \left(\frac{\lambda^s_{MC}(G_{mean}, C_{mean}) - \lambda^{s,(0)}_{SG}}{\lambda^s_{MC}(G_{mean}, C_{mean})} \right)^2 \tag{7.13}$$

where $E_{0th,\lambda^s,SG}$ is the error between the s'th eigenvalue found by Monte Carlo in case the uncertain variables equal their respective means ($\lambda^s_{MC}(G_{mean}, C_{mean})$) and the zeroth order coefficients of the expansion of the eigenvalues ($\lambda^{s,(0)}_{SG}$). The errors should be close to zero.

7.5.1 Uncertain shear modulus

First, the problem in case only the shear modulus is uncertain is considered. The results of the first validation are shown in Table 7.9, which are the mean errors of the leave-one-out cross validation for the first four different eigenvalues for different orders of the expansion of the stiffness matrix α and the eigenmodes and eigenvalues β and γ. All obtained errors are small and the results of the straight forward and the inexact method are similar. When using the straight forward method, the errors significantly decrease in case the expansion orders used for all variables is two or higher. For the inexact method, the same trend can only be seen for the first eigenvalues. For the remaining eigenvalues, the error changes little when varying the orders of the PC expansions. Overall, the smallest errors are obtained when the orders 3, 2 and 2 are used for stiffness matrix, modes and eigenvalues respectively, which will therefore be used in the following validations.

		Straight forward				**Inexact**			
$\boldsymbol{\alpha}$	$\boldsymbol{\beta = \gamma}$	$\boldsymbol{E_{\lambda_1}}$	$\boldsymbol{E_{\lambda_5}}$	$\boldsymbol{E_{\lambda_9}}$	$\boldsymbol{E_{\lambda_{11}}}$	$\boldsymbol{E_{\lambda_1}}$	$\boldsymbol{E_{\lambda_5}}$	$\boldsymbol{E_{\lambda_9}}$	$\boldsymbol{E_{\lambda_{11}}}$
0 1	0 1	8.75e-10	8.75e-10	8.75e-10	2.16e-9	8.75e-10	8.75e-10	8.75e-10	2.16e-09
0 1 2	0 1	4.47e-9	4.47e-9	4.47e-9	1.11e-8	4.47e-9	4.47e-9	4.47e-9	1.11e-8
0 1 2 3	0 1	1.89e-9	1.89e-9	1.89e-9	4.67e-9	1.89e-9	1.89e-9	1.89e-9	4.67e-9
0 1	0 1 2	5.21e-9	5.21e-9	5.21e-9	1.29e-8	5.21e-9	5.21e-9	5.21e-9	1.29e-8
0 1 2	0 1 2	1.11e-13	1.11e-13	1.11e-13	1.57e-12	1.11e-13	3.97e-9	1.31e-9	3.96e-9
0 1 2 3	0 1 2	2.90e-14	2.90e-14	2.90e-14	4.22e-13	2.90e-14	8.35e-9	2.14e-9	4.13e-9

Table 7.9: The results of the leave-one-out cross validation on expansions of different orders for the first 4 different eigenvalues, when the shear modulus is uncertain.

The result of the second validation for the first four different eigenvalues is shown in Figure 7.11. For all four eigenvalues, the distributions obtained through Monte Carlo, the straight forward method and the inexact method are almost perfectly aligned, indicating the mean and standard deviation of the distributions of the four eigenvalues considered are about equal for each method. The expansions obtained by application of stochastic Galerkin through the straight forward and the inexact method both approximate the eigenvalues of the original problem well.

The results of the third validation "zeroth order coefficients" are shown in Table 7.10. The errors are only shown for the first five different eigenvalues and for the average error of all 34 calculated eigenvalues. The errors of the first five eigenvalues are small and about equal for the two solution methods. The average error, however, is significantly smaller when applying the inexact method. The results by the straight forward method show a sudden rise in error for eigenvalues 19, 20, 33 and 34, of which the highest error is 9.25e-5 for the 19th eigenvalue. The second evaluation is applied on the 19th eigenvalue to investigate whether the rise in error has sigificant effect on the approximation of the distribution. The result is shown in Figure 7.12. The distributions are almost indistinguishable, indicating that, even though the mean by straight forward is less accurate compared to the inexact method, both approximate the mean and variance well. So, the distributions of the eigenvalues obtained by both the inexact and the straight forward method approximate the distributions obtained by Monte Carlo closely.

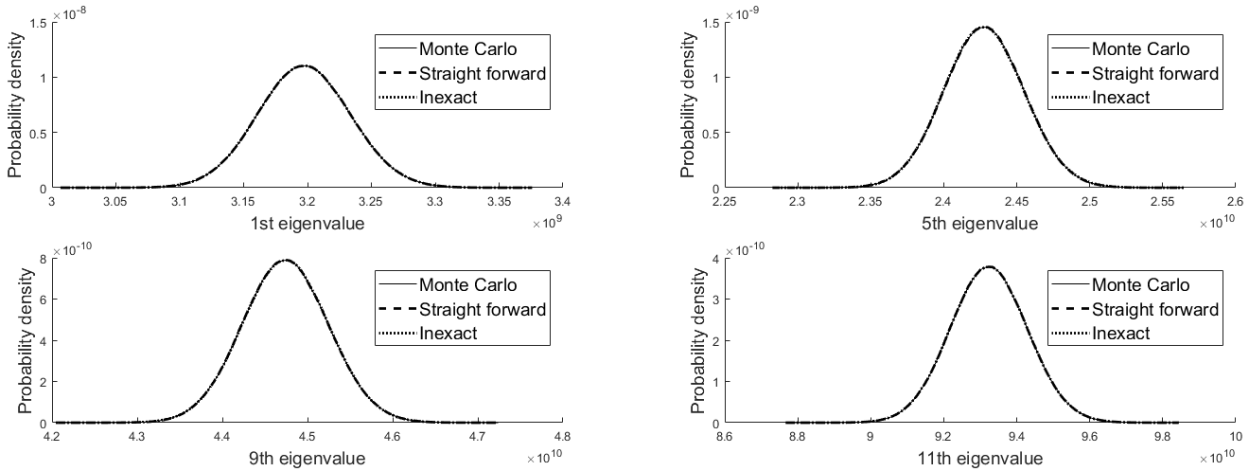


Figure 7.11: The distributions of the first four different eigenvalues based on 1e6 samples, when the shear modulus is uncertain.

	Straight forward	**Inexact**
E_{0th,λ_1}	3.81e-10	3.81e-10
E_{0th,λ_5}	3.81e-10	3.81e-10
E_{0th,λ_9}	3.81e-10	3.81e-10
$E_{0th,\lambda_{11}}$	5.19e-18	5.01e-18
$E_{0th,\lambda_{13}}$	3.81e-10	3.81e-10
Mean	8.73e-05	2.92e-10

Table 7.10: The result of the "zeroth order coefficients" validation for the first five different eigenvalues and the mean of all computed eigenvalues, when the shear modulus is uncertain.

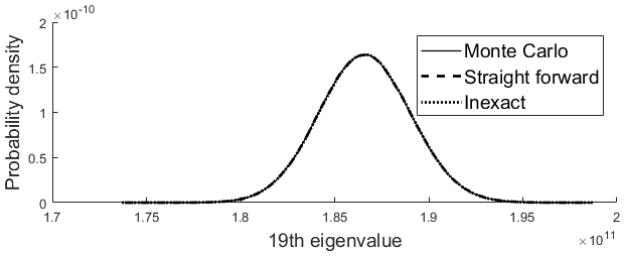


Figure 7.12: The distributions of the 19th eigenvalues based on 1e6 samples, when the shear modulus is uncertain.

Apart from the eigenvalues, the internal modes must be approximated well given the modes are used to simulate the flexible behaviour of dynamic systems. Figure 7.13 shows the first 16 eigenmodes determined for a random sample of G by Monte Carlo, the straight forward method and the inexact method. The figure clearly shows the shapes of the modes determined by the different methods are very similar, as is desired. The size of the deformations differs, as well as the direction. In the figure, the size of the deformations has been altered such that all modes are visible. The difference in size and direction is not a problem however, because the flexible coordinates compensate. The most important difference between the results is that for each Monte Carlo mode only one half of the bar is deformed, which is not true for the modes determined by stochastic Galerkin. The later have small deformations on the side of the bar which should not be deformed. However, the deformations are relatively small and will therefore likely not lead to problems. If problems are encountered, the deformations can be set to zero by hand.

So, both the straight forward and the inexact method approximate the eigenvalues well compared to Monte Carlo. The inexact method is a more reliable, since the straight forward method yields lesser results for some eigenvalues. However, the errors are still small enough to approximate the constrained eigenvalues closely. The shapes of the eigenmodes determined through both methods have a similar shape compared to the modes determined through Monte Carlo. However, for each mode one half of the bar should have no deflection, which is not the case for the results by stochastic Galerkin. The undesired deflections are small, but can be forced to zero to improve the results.

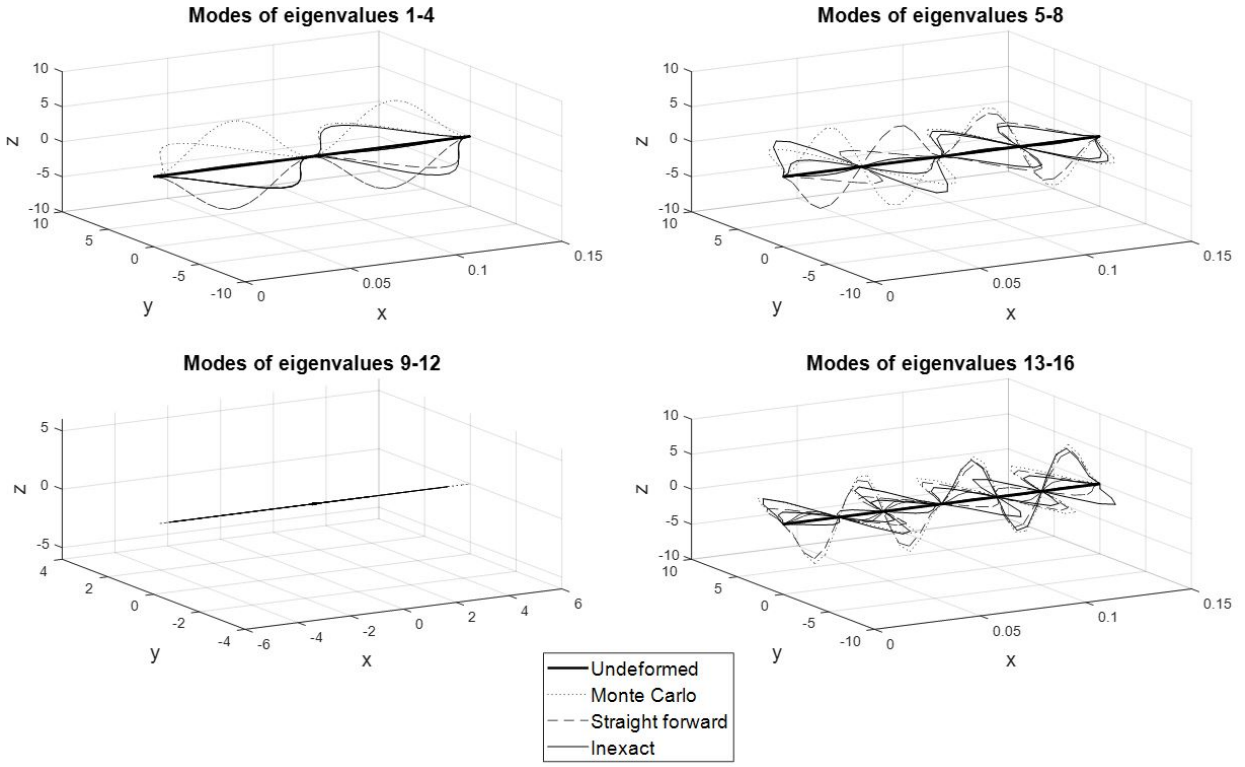


Figure 7.13: The first 16 eigenmodes determined by Monte Carlo, the straight forward method and the inexact method for a random sample of G.

7.5.2 Uncertain shear and bulk modulus

The problem in case both the shear and the bulk modulus are considered uncertain variables is discussed now. Table 7.11 shows the results of the first validation, which are the mean errors of the leave-one-out cross validation for the first four different eigenvalues. All obtained errors are small. For the straight forward method, the error decreases when orders higher than one are used for the stiffness matrix, while the orders for the eigenvalues and modes influence the error little. For the inexact method the opposite is the case: application of order two for the eigenvalues and modes consistently results in lower errors compared to order one. The order for the stiffness matrix however, has little effect on the error. For both the straight forward and the inexact method, using orders 2, 2 and 2 for the stiffness matrix, eigenmodes and eigenvalues respectively results in the smallest errors and will therefore be used as expansion orders in the remaining validations.

		Straight forward				**Inexact**			
$\boldsymbol{\alpha}$	$\boldsymbol{\beta = \gamma}$	$\boldsymbol{E_{\lambda_1}}$	$\boldsymbol{E_{\lambda_5}}$	$\boldsymbol{E_{\lambda_9}}$	$\boldsymbol{E_{\lambda_{11}}}$	$\boldsymbol{E_{\lambda_1}}$	$\boldsymbol{E_{\lambda_5}}$	$\boldsymbol{E_{\lambda_9}}$	$\boldsymbol{E_{\lambda_{11}}}$
0 1	0 1	5.59e-7	5.59e-7	5.59e-7	2.29e-9	5.59e-7	5.59e-7	5.59e-7	2.29e-9
0 1 2	0 1	4.75e-9	4.75e-9	4.75e-9	8.24e-9	5.52e-7	5.52e-7	5.52e-7	8.23e-9
0 1 2 3	0 1	3.10e-9	3.10e-9	3.10e-9	6.60e-9	4.80e-7	4.80e-7	4.80e-7	6.60e-9
0 1	0 1 2	2.69e-7	2.69e-7	2.69e-7	1.03e-8	9.17e-9	9.17e-9	9.17e-9	1.45e-8
0 1 2	0 1 2	1.77e-9	1.77e-9	1.77e-9	4.18e-9	1.77e-9	1.77e-9	1.77e-9	4.18e-9
0 1 2 3	0 1 2	1.21e-9	1.21e-9	1.21e-9	3.58e-9	5.00e-9	5.00e-9	5.00e-9	1.42e-8

Table 7.11: The results of the leave-one-out cross validation on expansions of different orders for the first 4 different eigenvalues, when the shear and bulk modulus are uncertain.

Figure 7.14 shows the results of the second evaluation for the first four different eigenvalues. The graphs all are almost perfectly aligned, indicating the means and variances found through the straight forward and the inexact methods are about equal to the means and variances according to Monte Carlo. So, both applications of stochastic Galerkin are able to approximate the result of Monte Carlo.

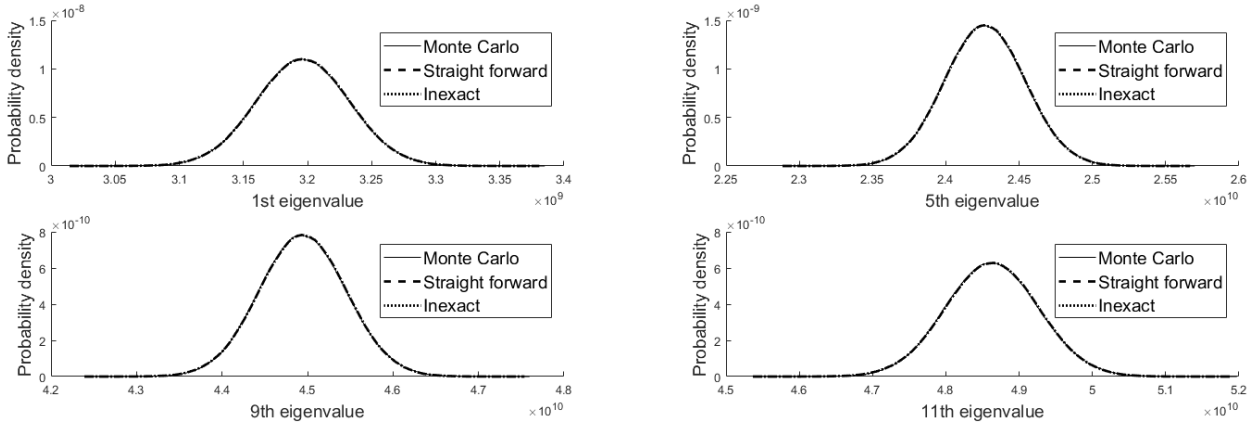


Figure 7.14: The distributions of the first four different eigenvalues based on 1e6 samples, when the shear and bulk modulus are uncertain.

The results of the third validation for the first five different eigenvalues and the average of all 28 calculated eigenvalues are given in Table 7.12. All errors are small for both methods, meaning the means of the eigenvalues are approximated well compared to Monte Carlo, confirming the results of the previous validation. When repeating the calculations, the average error using the straight forward method can have values in the range 1e-5, as was the case when only the shear modulus was uncertain. However, the error is still small enough for the distributions of the eigenvalues to approximate the distributions by Monte Carlo closely.

	Straight forward	**Inexact**
E_{0th,λ_1}	5.64e-10	5.64e-10
E_{0th,λ_5}	5.64e-10	5.64e-10
E_{0th,λ_9}	5.64e-10	5.64e-10
$E_{0th,\lambda_{11}}$	6.01e-15	6.02e-15
$E_{0th,\lambda_{13}}$	5.64e-10	5.64e-10
Mean	4.03e-10	4.14e-10

Table 7.12: The result of the "zeroth order coefficients" validation for the first five different eigenvalues and the mean of all computed eigenvalues, when the shear and bulk modulus are uncertain.

Because the eigenvalues are approximated closely by the expansions, the eigenmodes should be considered as well. Figure 7.15 shows the first 16 internal modes for random samples of G and C. The result is similar to the result found when only G was varied: the same deformation shapes are found by each method, but the directions and sizes of the deformations vary. Furthermore, the half of the bar which has zero deflection for Monte Carlo has small deflections for both the straight forward and the inexact methods. The direction and size of the deformation is compensated by the flexible coordinates and is thus not a problem. However, the zero deflection might have to be enforced artificially to improve the results of the dynamic analysis.

In conclusion, compared to Monte Carlo both the straight forward and the inexact method approximate the eigenvalues well. However, the straight forward method is less reliable, since occasionally the error for some eigenvalues increases compared to the inexact method. Because the errors are small, the straight forward method still yields good results regardless. The internal Craig Bampton modes determined by each method have similar shapes. The direction and size of the deformations differs, which is not a problem. However, the modes determined by the straight forward and the inexact methods have small deflections at the side of the bar which should have none. The deflections can be forced to zero by hand to improve the results of the dynamic analysis if necessary.

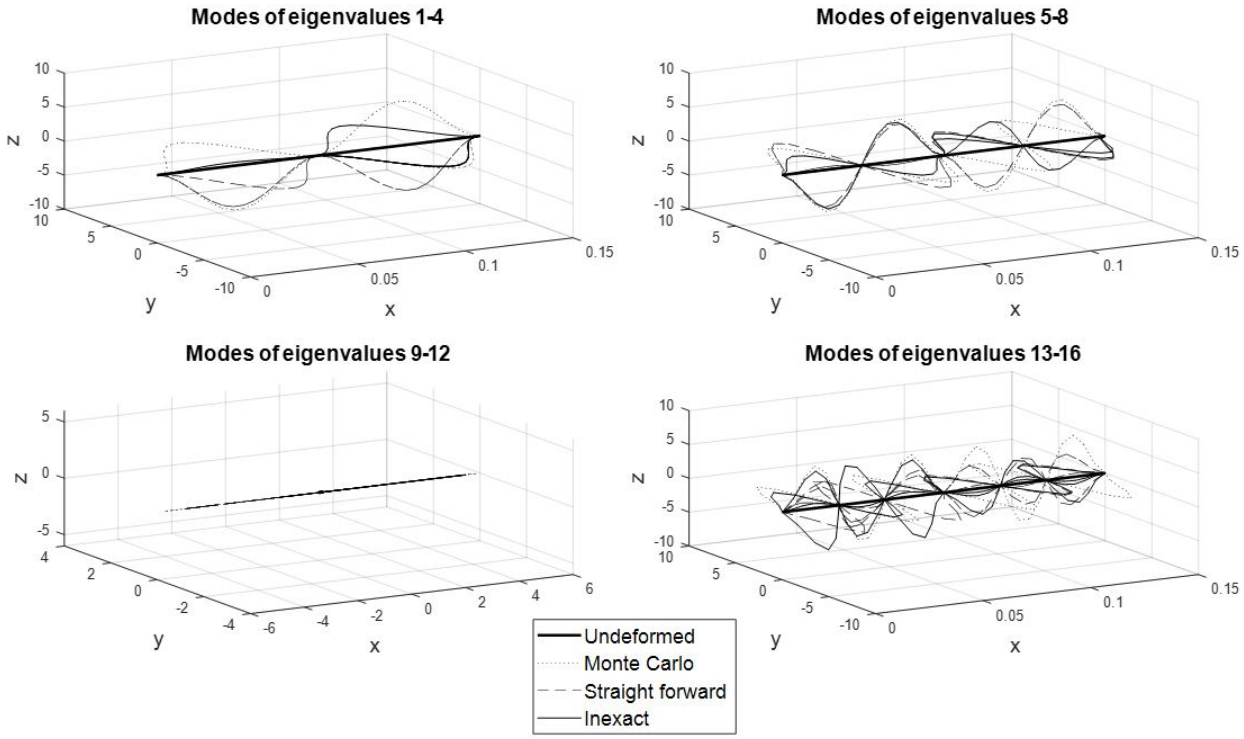


Figure 7.15: The first 16 eigenmodes determined by Monte Carlo, the straight forward method and the inexact method for a random sample of G and C.

7.5.3 Imaginary results

A problem which arose during the application of stochastic Galerkin, both when using the straight forward and the inexact method, is the presence of imaginary results when solving the eigenvalue problem using the MATLAB routine "eig()". As a solution, the routine "eigs()" was used instead which allows the user to only compute the first n eigenvalues and vectors. If n was set to 100 while using "eig()" returns 108 eigenvalues and modes, imaginary results are much less frequently encountered, suggesting problem is ill-posed. However, the reason why the imaginary results appear is unclear. To prevent imaginary numbers from entering the calculations, every solution of the eigenvalue problem is confirmed to only contain real values. If not, the problem is solved again until real results are obtained.

To investigate the imaginary results, the eigenvalue problem is solved several times with different settings. The problem is solved in the original generalized shape in MATLAB by "eig($\boldsymbol{K}_{ii}$,$\boldsymbol{M}_{ii}$)" and in the standard shape using Cholesky factorization by "eig($\boldsymbol{L}^{-1}\boldsymbol{K}_{ii}\boldsymbol{L}^{-T}$)". In both shapes, the solution is be obtained using the routines "eig()" and "eigs()" for a different percentage of the eigenvalues, where 100% is the number of eigenvalues obtained when solving the problem using "eig()". Finally, each of the solution methods is applied to the bar simulated with a different number of nodes. For every case, the solution is determined ten times and the number of times imaginary results are found is counted. Table 7.13 shows the results.

Problem shape		Original					Cholesky				
Computed λ's		100%	100%	80%	50%	20%	100%	100%	80%	50%	20%
Routine		eig()	eigs()	eigs()	eigs()	eigs()	eig()	eigs()	eigs()	eigs()	eigs()
Number of nodes	5	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	10	10	1	0	0
	21	0	0	0	0	0	10	10	3	0	0
	31	0	0	0	0	0	10	10	6	3	1
	41	0	0	0	0	0	10	10	7	7	8

Table 7.13: The number of times imaginary results are found if the constrained eigenvalue problem is solved ten times using the method and settings defined in the table.

The table shows the imaginary results appear only when the generalized problem is transformed to the standard shape. When only the first eigenvalues and modes are determined using the standard form, imaginary results do not occur either. However, the more eigenvalues considered, the higher the change of encountering imaginary results. It is evident in which cases the imaginary results are likely to appear and thus that the practical solution of calculating only the first eigenvalues is effective, however why the problem becomes ill-posed is not clear.

8 Discussion

The internal Craig Bampton modes have been defined as the constrained eigenmodes of half of the bar such that the node in the center of mass is never displaced. Alternatively, the internal modes can be defined as the constrained eigenmodes of the complete bar. Because the boundary nodes are stationary in all constrained eigenmodes, rigid body modes cannot be formed regardless of whether the node in the center of mass is stationary as well. The use of the eigenmodes of the complete bar is expected to be more efficient compared the use of the eigenmodes of half of the bar in terms of the number of internal modes required to achieve a simulation of the flexible behaviour of the internal nodes with a similar level of detail.

The quadratic velocity terms in the simulation of the slider crank are shown to depend on the driving torque and whether the torque is harmonic or constant. However, it is also expected that the simulation time and material properties such as mass have an influence on the size of the velocities and thus on the size of the quadratic velocity terms. Therefore, it might be interesting to experiment more with the different simulation properties of the slider crank to investigate when the quadratic velocity terms cease to be neglectable.

When reducing the simulation of the slider crank by POD, it is found that the number of samples in the snapshot matrix and the truncation order of the POD basis influence the quality of the reduction. However, only four settings for both parameters have been considered. It might be interesting to experiment with more values to compare the results for more extreme settings and to estimate the optimal values for the parameters more precisely. Furthermore, the best performing POD basis can be tested further by increasing the simulation time until the error between the unreduced and POD reduced simulations start to deviate from the error between the unreduced and the Craig Bampton reduced simulations. The moment the error starts to grow large, the POD basis might be replaced such that the error is reduced to an acceptable level again. Furthermore, the reduction order of the bases of the two bars have been chosen equal every time. Choosing the reduction order separately might lead to further reduction. Finally, the POD reduction should also be applied on a more complex model with more variables and more significant flexible behaviour to examine whether POD performs equally well on more complex dynamic structures.

PC expansions have been made for the x-position of point A of the slider crank at 0.1s with as uncertain input parameters either the shear or the shear and bulk modulus. The expansions result in small errors, indicating the expansions are good approximations of the original model. However, the quality of the metamodels can be examined further by applying Monte Carlo on the expansions for a million samples and comparing the result to application of Monte Carlo on the unreduced model. The comparison has not been done in this report due to time constraints, since applying Monte Carlo on the original model would take much time. Furthermore, application of PCE to acquire metamodels for the remaining coordinates, velocities and accelerations of the slider crank should be made.

When applying POD on the Craig Bampton modes in case of an uncertain shear modulus, the approximations of the internal modes yield good results under the condition that the number of samples in the snapshot matrix is larger than the truncation order. The condition is obtained from very few results and should therefore be examined more thoroughly before one can say with certainty that the condition holds.

Stochastic Galerkin has been applied on the Craig Bampton internal modes and corresponding eigenvalues to obtain expansions for the variables which approximate the original problem well. However, during the calculation of the expansions, the generalized eigenvalue problem is transformed to a standard shape through Cholesky factorization. Solving the transformed problem frequently results in imaginary numbers which were not expected. In practise, the problem could be solved by computing only the first eigenvalues and eigenvectors. However, why the problem becomes ill-posed, resulting in imaginary numbers, is not clear. Further investigation should be done to find the cause of the problem and how to prevent it.

9 Conclusions and recommendations

The movement of the three-dimensional model of the slider crank can be simulated with sufficient detail when applying only Craig Bampton boundary modes to reduce the model and when the DAE is transformed to an ODE and integrated over time by Runge-Kutta. The model of the slider crank is driven by a torque of 1Nm and simulated over 0.25 seconds, in which case the quadratic velocity terms can be neglected. When a higher torque or a longer simulation time is used, the importance of the quadratic velocity terms needs to be re-evaluated since both cause a grow in velocities resulting in larger quadratic velocity terms.

Using Craig Bampton modes to model the slider crank is a method to achieve reduction of the number of generalized flexible coordinates, raising the question whether the data-driven technique POD could be applied on the unreduced simulation of the slider crank to achieve a similar or larger reduction compared to Craig Bampton, under the condition that the errors between the unreduced and reduced simulations are similar. Reducing the rigid motion by POD yields poor results, probably due to the strong non-linearities in the rigid movement. By applying POD on the flexible movement of the two bars separately and tuning the number of samples in the snapshot matrix and the training time, the simulation is reduced beyond what is achieved by Craig Bampton: a POD basis consisting of two times six rigid modes and two times three flexible modes. Applying Craig Bampton on the slider crank requires a basis of two times six rigid modes and two times twelve flexible modes. Using POD, a result with a similar error compared to Craig Bampton was achieved while using only halve the number of modes. By investigating more possible settings for the truncation order and the number of samples and considering the truncation orders of the bases of both bars separately, further reduction might be achieved still. Because the current settings already yield good results for the slider crank, it is interesting to investigate how well POD performs on a more complicated model by applying POD on the simulation of a structure with a larger number of generalized coordinates and more significant flexible behaviour.

In case the shear and bulk modulus of the slider crank are uncertain, the result of the simulation depends on the uncertain variables. To find the distribution of the result, a PC expansion was made which was to approximate the original model well and reduce the evaluation time such that Monte Carlo became feasible. Two metamodels have been made of which the x-position of point A of the slider crank after 0.1s is the output variable and either the shear or the shear and bulk modulus are the random input variables. For the first case, the best approximation is achieved when a first order expansion is used, resulting a leave-one-out cross validation error in the range 8e-9. For the second case, a third order expansion gives the best result, with an error in the range 6e-8. Because both yield a small error, both are good approximations of the original model and both reduce the computation time significantly, meaning Monte Carlo can be applied on the expansions for a large number of samples within an acceptable time frame. In future research, Monte Carlo can be applied to the expansions and compared to the result of Monte Carlo on the original model. By comparing the resulting distributions, specifically the tails, the quality of the expansion can be assessed further.

The Craig Bampton boundary and internal modes of the slider crank are dependent on the shear modulus. POD has been applied to discretize the stochastic space with good results. The boundary modes are affected little by changes in the shear modulus, meaning that application of POD on the boundary modes results in a small error (range 1e-30) regardless of the number of samples in the snapshot matrix and the truncation order of the basis. However, using more POD modes does decrease the error further, adding the detailed influence of the shear modulus on the modes to the basis. Because the error is low, it is concluded that the Craig Bampton boundary modes are linear combinations of the POD modes. The constrained eigenmodes and eigenvalues are more strongly dependent on the shear modulus, resulting in larger errors. However, the errors are still small (range 1e-23) under the condition that the number of samples in the snapshot matrix is larger than the truncation order. The condition is determined from a limited number of results and should thus be investigated further. The small errors lead to the conclusion that the constrained eigenmodes are linear combinations of the POD modes as well.

Because the constrained eigenmodes depend on the shear modulus, it is possible to obtain a metamodel of the internal modes and corresponding eigenvalues using stochastic Galerkin of which the evaluation time is reduced compared to the deterministic model such that application of Monte Carlo on the metamodel to estimate the distributions of the modes and eigenvalues is feasible. The results in case either the shear modulus or both the shear and bulk modulus are considered uncertain are very similar. The best performing

metamodels are obtained when using a second order expansion for the eigenvalues and modes and a second or third order expansion for the stiffness matrix. When only the shear modulus is considered, the leave-one-out cross validation error is smaller when using the straight forward method (range 1e-14) compared to the inexact method (range 1e-9). When both the shear and bulk modulus are uncertain, the errors are in the same range for both techniques (range 1e-9). However, when using the straight forward method, the means of some of the higher eigenvalues are estimated with a higher error (range 1e-5). All though the error is larger, the expansions of the eigenvalues still estimate the distributions of the respective eigenvalues closely. The modes obtained through the expansions have the expected shape, but the orientations and scaling factors differ. The difference in orientation and scaling is compensated by the generalized coordinates and thus not a problem. However, the modes obtained through stochastic Galerkin have small deflections at the side of the bar which should have none. If the non-zero deflection leads to problems during the FMBD simulations, the deflections can be forced to zero by hand. So, both the straight forward and inexact methods can be used to obtain metamodels of the constrained eigenmodes and values through stochastic Galerkin, on which Monte Carlo can be applied to approximate the distributions of the variables closely. During the application of stochastic Galarkin, imaginary results are encountered when performing eigenvalue decomposition on a problem which has been transformed from a generalized shape to a standard shape using Cholesky factorization. The imaginary results are due to the problem becoming ill-posed, but to find the cause further research is required. In practice, the problem was solved by calculating solely the lower eigenvalues and eigenmodes.

The reduction methods have all been applied to the flexible multibody dynamic model of the slider crank successfully, which means that application of data-driven reduction techniques to achieve uncertainty quantification on flexible multibody problems is feasible. Therefore, it is especially interesting to, in further research, try to apply the techniques described in this report on a more complex dynamic system. Especially a system which contains more significant flexible behaviour and a larger set of generalized coordinates.

References

[1] J. Schilder, "Dynamics 3, flexible multibody dynamics," Lecture notes, MS3, University of Twente, 2019.

[2] ——, "Flexible multibody dynamics, superelements using absolute interface coordinates in the floating frame formulation," PhD Dissertation, Univeristy of Twente, 2018.

[3] G. Stefanou, "The stochastic finite element method: Past, present and future," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 9, pp. 1031 – 1051, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782508004118

[4] R. Masoudi, T. Uchida, and J. McPhee, "Reduction of multibody dynamic models in automotive systems using the proper orthogonal decomposition," *Journal of Computational and Nonlinear Dynamics*, vol. 10, p. 031007, 05 2015.

[5] Y. Hou, C. Liu, and H. Hu, "Component-level proper orthogonal decomposition for flexible multibody systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112690, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782519305754

[6] B. Sudret, "Samo 2016: Polynomial chaos expansions in engineering, bruno sudret," https://www.youtube.com/watch?v=fxUVBSbmqdI, December 2016, recording of a lecture by Bruno Sudret at Université de La Réunion on Youtube.

[7] F. W. Shuxing Yang, Fenfen Xiong, "Polynomial chaos expansion for probabilistic uncertainty propagation," in *Uncertainty Quantification and Model Calibration*, 1st ed., J. P. Hessling, Ed. IntechOpen, July 2017, ch. Chapter 2.

[8] E. Torre, S. Marelli, P. Embrechts, and B. Sudret, "Data-driven polynomial chaos expansion for machine learning regression," *Journal of Computational Physics*, vol. 388, 03 2019.

[9] K. Lee and B. Sousedík, "Inexact methods for symmetric stochastic eigenvalue problems," vol. 6, pp. 1744–1776, 12 2018.

[10] H. C. Elman and T. Su, "Low-rank solution methods for stochastic eigenvalue problems," 2018.

[11] J. G. de Jalon and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*, E. F. G. Frederick F. Ling, Ed. Springer-Verlag, 1994.

[12] H. S. Chin, "Stabilization methods for simulations of constrained multibody dynamics," PhD Dissertation, The University of British Columbia, May 1995.

[13] J. Schilder, "Reader dynamics 3, flexible multibody dynamics," Lecture notes, MS3, University of Twente, 2017.

[14] MathWorks, "Ode45, solve nonstiff differential equations — medium order method," https://nl.mathworks.com/help/matlab/ref/ode45.html, 2020.

[15] S. Weiland, "Model reduction," Lecture sheets, Department of Electrical Engineering, Eindhoven University of Technology, 2018.

[16] C. Nowakowski, J. Fehr, M. Fischer, and P. Eberhard, "Model order reduction in elastic multibody systems using the floating frame of reference formulation," *IFAC Proceedings Volumes*, vol. 45, no. 2, pp. 40 – 48, 2012, 7th Vienna International Conference on Mathematical Modelling. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016306401

[17] C. Ceccato, X. Zhou, D. Pelessone, and G. Cusatis, "Proper orthogonal decomposition framework for the explicit solution of discrete systems with softening response," *Journal of Applied Mechanics*, vol. 85, 01 2018.

[18] G. Kerschen and J. Golinval, "Physical interpretation of the proper orthogonal modes using the singular value decomposition," *Journal of Sound and Vibration*, vol. 249, no. 5, pp. 849 – 865, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022460X01939306

[19] D. Xiu, *Numerical Methods for Stochastic Computations, A Spectral Method Approach*, 10th ed. 41 William Street, Princeton, New Jersey 08540: Princeton University Press, 2010.

[20] S. Blackford, "Generalized symmetric definite eigenproblems," https://www.netlib.org/lapack/lug/node54.html, October 1999.

[21] K. Y. Patarroyo, "A digression on hermite polynomials," February 2019.

[22] B. V. Rosić and J. H. Diekmann, "Methods for the uncertainty quantification of aircraft simulation models," *Journal of Aircraft*, vol. 52, no. 4, pp. 1247–1255, 2015. [Online]. Available: https://doi.org/10.2514/1.C032856

[23] E. Perdahcioğlu, T. Meinders, and A. van den Boogaard, "An introduction yo the finite element method," Lecture notes, Faculteit construerende technische wetenschappen, University of Twente, 2017.

Appendices

A Quadratic velocity terms

In the derivation of the constrained equation of motion, the quadratic velocity terms have been discarded because the quadratic velocity terms are typically small. However, if a problem with fast rotational movement is considered, the quadratic velocity terms will be significant and should be included in the constrained equation of motion. To include the quadratic terms, one must understand how the terms are assembled. The quadratic velocity terms are expressed as:

$$\boldsymbol{Q}_v = \begin{bmatrix} \begin{bmatrix} \boldsymbol{R}_1^O \end{bmatrix} \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{M}_{FE} \begin{bmatrix} \dot{\boldsymbol{q}}_{nodes}^2 \end{bmatrix} \tag{A.1}$$

where $\boldsymbol{Q}_v$ are the quadratic velocity terms, $\boldsymbol{R}_1^O$ are the rotation matrix of frame 1 with respect to frame O, $\boldsymbol{\Phi}_{rig}$ are the rigid modes, $\boldsymbol{\Phi}_{flex}$ are the flexible modes, $\boldsymbol{M}_{FE}$ is the FE mass matrix and $\dot{\boldsymbol{q}}_{nodes}$ is defined as:

$$\dot{\boldsymbol{q}}_{nodes} = \begin{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{r}_A^{11} + 2\tilde{\boldsymbol{\omega}}_1^{1O} \dot{\boldsymbol{r}}_A^{11} \\ \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{\omega}_A^{11} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{r}_N^{11} + 2\tilde{\boldsymbol{\omega}}_1^{1O} \dot{\boldsymbol{r}}_N^{11} \\ \tilde{\boldsymbol{\omega}}_1^{1O} \boldsymbol{\omega}_N^{11} \end{bmatrix} \end{bmatrix} \tag{A.2}$$

for a body with N nodes where $\tilde{\boldsymbol{\omega}}_1^{1O}$ is the skew matrix of the angular velocity vector of point 1 with respect to point O expressed in frame 1, $\boldsymbol{r}_A^{11}$ is the position vector of point A with respect to point 1 expressed in frame 1, $\dot{\boldsymbol{r}}_A^{11}$ is the velocity vector of point A with respect to point 1 expressed in frame 1, $\boldsymbol{\omega}_A^{11}$ is the angular velocity vector of point A with respect to point 1 expressed in frame 1. One can write (A.2) in matrix vector form by separating the skew matrices and the vectors and substituting the result in (A.1) to obtain:

$$\boldsymbol{Q}_v = \begin{bmatrix} \begin{bmatrix} \boldsymbol{R}_1^O \end{bmatrix} \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{M}_{FE} \begin{bmatrix} \begin{bmatrix} 2\tilde{\boldsymbol{\omega}}_1^{1O} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{\omega}}_1^{1O} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{r}}_A^{11} \\ \boldsymbol{\omega}_A^{11} \end{bmatrix} + \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_A^{11} \\ \boldsymbol{0} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} 2\tilde{\boldsymbol{\omega}}_1^{1O} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{\omega}}_1^{1O} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{r}}_N^{11} \\ \boldsymbol{\omega}_N^{11} \end{bmatrix} + \begin{bmatrix} \tilde{\boldsymbol{\omega}}_1^{1O} \tilde{\boldsymbol{\omega}}_1^{1O} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_N^{11} \\ \boldsymbol{0} \end{bmatrix} \end{bmatrix} \tag{A.3}$$

Because the (angular) acceleration vector and position vector describe the movement of the nodes on the body with respect to the floating frame, the vectors can be written in terms of the flexible modes and flexible generalized coordinates. The skew matrices of the angular velocities can be expressed in terms of the global frame using rotation matrices:

$$\boldsymbol{Q}_v = \begin{bmatrix} \begin{bmatrix} \boldsymbol{R}_1^O \end{bmatrix} \boldsymbol{\Phi}_{rig}^T \\ \boldsymbol{\Phi}_{flex}^T \end{bmatrix} \boldsymbol{M}_{FE} \begin{bmatrix} \begin{bmatrix} 2\boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \end{bmatrix} \boldsymbol{\Phi}_{A,flex} \dot{\boldsymbol{\eta}}_A + \begin{bmatrix} \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Phi}_{A,flex} \boldsymbol{\eta}_A \\ \vdots \\ \begin{bmatrix} 2\boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \end{bmatrix} \boldsymbol{\Phi}_{N,flex} \dot{\boldsymbol{\eta}}_N + \begin{bmatrix} \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\Phi}_{N,flex} \boldsymbol{\eta}_N \end{bmatrix} \tag{A.4}$$

where $\boldsymbol{R}_O^1$ is the rotation matrix of frame O with respect to frame 1, $\boldsymbol{\Phi}_{A,flex}$ are the flexible modes at node A, $\boldsymbol{\eta}_A$ is the vector of flexible generalized coordinates of point A and $\dot{\boldsymbol{\eta}}_A$ is the vector of flexible generalized velocities of point A.

B Computation of the POD basis

The POD basis can be found by either performing singular value decomposition on the centralized snapshot matrix:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{B.1}$$

where $\boldsymbol{U}$ is a matrix containing the left-singular vectors, $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values and $\boldsymbol{V}$ is a matrix containing the right-singular vectors, or by eigenvalue decomposition of the covariance matrix of the snapshot matrix:

$$\mathrm{cov}(\boldsymbol{S}_0) = \boldsymbol{U}\sqrt{\boldsymbol{\Sigma}}\boldsymbol{U}^{-1} \tag{B.2}$$

where $\mathrm{cov}(\boldsymbol{S}_0)$ is the covariance matrix of the non-centralized snapshot matrix, $\boldsymbol{U}$ is a matrix containing the eigenvectors and $\sqrt{\boldsymbol{\Sigma}}$ is a diagonal matrix containing the eigenvalues [4, 5, 15, 16]. Both methods result in the same POD basis $\boldsymbol{U}$. To understand why, the relation between SVD and eigenvalue decomposition and the relation between the covariance and the snapshot matrix must be understood.

B.1 The relation between SVD and eigenvalue decomposition

The standard eigenvalue problem of matrix $\boldsymbol{A}$ is defined as:

$$\boldsymbol{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{B.3}$$

where $\boldsymbol{v}_i$ are the eigenvectors of matrix $\boldsymbol{A}$ corresponding to the eigenvalues λ_i. Alternatively, the eigendecomposition of a matrix $\boldsymbol{A}$ is defined as:

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \tag{B.4}$$

where every column of $\boldsymbol{Q}$ is an eigenvector $\boldsymbol{v}_i$ and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_i. The i'th eigenvalue corresponds to the i'th eigenvector.

The SVD of matrix $\boldsymbol{M}$ is defined as

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* \tag{B.5}$$

where the columns of $\boldsymbol{U}$ contain the left singular vectors $\boldsymbol{u}_i$ and the columns of $\boldsymbol{V}$ contain the right singular vectors $\boldsymbol{v}_i$ of $\boldsymbol{M}$. $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values σ_i. Again, the i'th singular value corresponds to the i'th singular vectors.

The relation between the decompositions becomes apparent by multiplying the SVD of matrix $\boldsymbol{M}$ with its conjugate transposed $\boldsymbol{M}^*$:

$$\boldsymbol{M}\boldsymbol{M}^* = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\boldsymbol{V}\boldsymbol{\Sigma}^*\boldsymbol{U}^* = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^*\boldsymbol{U}^* \tag{B.6}$$

or by pre-multiplying the SVD of $\boldsymbol{M}$ with $\boldsymbol{M}^*$:

$$\boldsymbol{M}^*\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Sigma}^*\boldsymbol{U}^*\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \boldsymbol{V}\boldsymbol{\Sigma}^*\boldsymbol{\Sigma}\boldsymbol{V}^* \tag{B.7}$$

The shape of the eigenvalue decomposition can be recognized in both formulations (B.6) and (B.7), where the left singular vectors $\boldsymbol{U}$ of $\boldsymbol{M}$ are the eigenvectors of $\boldsymbol{M}\boldsymbol{M}^*$ and the right singular vectors $\boldsymbol{V}$ of $\boldsymbol{M}$ are the eigenvectors of $\boldsymbol{M}^*\boldsymbol{M}$. The singular values $\boldsymbol{\Sigma}$ of $\boldsymbol{M}$ are the square root of the eigenvalues of $\boldsymbol{M}\boldsymbol{M}^*$ and $\boldsymbol{M}^*\boldsymbol{M}$.

B.2 The relation between the covariance and the snapshot matrix

The covariance matrix of matrix $\boldsymbol{A}$ is defined as:

$$\boldsymbol{C} = E[(\boldsymbol{A} - E[\boldsymbol{A}])(\boldsymbol{A} - E[\boldsymbol{A}])] \tag{B.8}$$

where $\boldsymbol{C}$ is the covariance matrix of $\boldsymbol{A}$. If $\boldsymbol{A}$ is shaped as the snapshot matrix (every row being one variable and every column being one time instance) the expected values can be written as:

$$\boldsymbol{C} = \frac{1}{n}(\boldsymbol{A} - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{A}_{ij})(\boldsymbol{A} - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{A}_{ij})^T \tag{B.9}$$

By defining the expected value of matrix $\boldsymbol{A}$ as:

$$\bar{\boldsymbol{A}} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{A}_{ij} \tag{B.10}$$

One can write (B.9) more compactly:

$$\boldsymbol{C} = \frac{1}{n}(\boldsymbol{A} - \bar{\boldsymbol{A}})(\boldsymbol{A} - \bar{\boldsymbol{A}})^T \tag{B.11}$$

The covariance of the snapshot matrix is thus defined as:

$$\mathrm{cov}(\boldsymbol{S}_0) = \frac{1}{N}(\boldsymbol{S} - \bar{\boldsymbol{S}})(\boldsymbol{S} - \bar{\boldsymbol{S}})^T = \frac{1}{N}\boldsymbol{S}\boldsymbol{S}^T \tag{B.12}$$

where N is the number of samples and the snapshot matrix contains only real values.

From the relation between eigenvalue decomposition and SVD (B.6), it is known that SVD of the centralized snapshot matrix can be written as:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{B.13}$$

and the eigenvalue decomposition of $\boldsymbol{S}\boldsymbol{S}^T$ as:

$$\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{U}\sqrt{\boldsymbol{\Sigma}}\boldsymbol{U}^T \tag{B.14}$$

thus yielding the same POD basis $\boldsymbol{U}$. However, $\boldsymbol{S}\boldsymbol{S}^T$ is not equal to the covariance matrix of $\boldsymbol{S}_0$. Even so, multiplying a matrix with a coefficient and applying eigenvalue decomposition on that matrix only influences the eigenvalues, not the eigenvectors:

$$(a\boldsymbol{A})\boldsymbol{v} = a(\boldsymbol{A}\boldsymbol{v}) = a(\lambda\boldsymbol{v}) = (a\lambda)\boldsymbol{v} \tag{B.15}$$

where $\boldsymbol{v} \neq \boldsymbol{0}$ is an eigenvector of $\boldsymbol{A}$ corresponding to the eigenvalue λ and a is some scalar. If λ is an eigenvalue of $\boldsymbol{A}$, then $a\lambda$ is an eigenvalue of $a\boldsymbol{A}$ and if $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{A}$ then $\boldsymbol{v}$ is also an eigenvector of $a\boldsymbol{A}$.

So, the eigenvalue decomposition of the covariance matrix of the snapshot matrix yields a matrix of eigenvectors which is equal to the matrix of left-singular vectors of the centralized snapshot matrix, proving why both methods for calculating the basis give the same result.

C Delta tensor

The expected value of the product of three orthonormal polynomials is equal to the delta tensor:

$$\mathbb{E}[\Psi_\alpha(\xi)\Psi_\beta(\xi)\Psi_\gamma(\xi)] = c_{\beta\gamma}^{(\alpha)} \tag{C.1}$$

where $\Psi_\alpha(\xi)$, $\Psi_\beta(\xi)$ and $\Psi_\gamma(\xi)$ are three orthonormal polynomials depending on the uncertain variable ξ and $c_{\beta\gamma}^{(\alpha)}$ is the delta tensor [21]. To understand why (C.1) holds, first the product of two orthonormal polynomials must be examined. The product of two polynomials of the same type of some orders n and m can also be written as a sum of the same type of polynomials up to order $m+n$ where every polynomial is multiplied with a coefficient:

$$\Psi_m(\xi)\Psi_n(\xi) = \sum_{l=0}^{m+n} c_{mn}^{(l)}\Psi_l(\xi) \tag{C.2}$$

One can use (C.2) to simplify the product of three orthonormal polynomials:

$$\mathbb{E}[\Psi_\alpha(\xi)\Psi_\beta(\xi)\Psi_\gamma(\xi)] = \mathbb{E}[\Psi_\alpha(\xi)\sum_{\epsilon=0}^{\beta+\gamma} c_{\beta\gamma}^{(\epsilon)}\Psi_\epsilon(\xi)] \tag{C.3}$$

The constant delta tensor can be taken out of the expectation to obtain:

$$\mathbb{E}[\Psi_\alpha(\xi)\Psi_\beta(\xi)\Psi_\gamma(\xi)] = \sum_{\epsilon=0}^{\beta+\gamma} c_{\beta\gamma}^{(\epsilon)}\mathbb{E}[\Psi_\alpha(\xi)\Psi_\epsilon(\xi)] \tag{C.4}$$

Because the polynomials are orthonormal, the expectation of two polynomials equals the Kronecker delta $\delta_{\alpha\epsilon}$:

$$\mathbb{E}[\Psi_\alpha(\xi)\Psi_\beta(\xi)\Psi_\gamma(\xi)] = \sum_{\epsilon=0}^{\beta+\gamma} c_{\beta\gamma}^{(\epsilon)}\delta_{\alpha\epsilon} = c_{\beta\gamma}^{(\alpha)} \tag{C.5}$$

Therefore, the delta tensor consists of the coefficients in (C.2).

The following example illustrate how the coefficients are defined. The product of two Hermite polynomials of order two equals:

$$\mathrm{He}_2(\xi)\mathrm{He}_2(\xi) = 16\xi^4 - 16\xi^2 + 4 \tag{C.6}$$

where He_n is the probabilists' Hermite polynomial of order n. The product of the two Hermite polynomials can be written as the sum of Hermite polynomials up to order four:

$$\mathrm{He}_2(\xi)\mathrm{He}_2(\xi) = c_{22}^{(0)}\mathrm{He}_0(\xi) + c_{22}^{(1)}\mathrm{He}_1(\xi) + c_{22}^{(2)}\mathrm{He}_2(\xi) + c_{22}^{(3)}\mathrm{He}_3(\xi) + c_{22}^{(4)}\mathrm{He}_4(\xi) \tag{C.7}$$

Substitution of the expression of the polynomials gives:

$$16\xi^4 - 16\xi^2 + 4 = c_{22}^{(0)}1 + c_{22}^{(1)}2\xi + c_{22}^{(2)}(4\xi^2 - 2) + c_{22}^{(3)}(8\xi^3 - 12\xi) + c_{22}^{(4)}(16\xi^4 - 48\xi^2 + 12) \tag{C.8}$$

from which the values of the delta tensors can be derived as:

$$16\xi^4 - 16\xi^2 + 4 = 8\cdot 1 + 0\cdot 2\zeta + 8(4\xi^2 - 2) + 0(8\xi^3 - 12\xi) + 1(16\xi^4 - 48\xi^2 + 12) \tag{C.9}$$

An efficient method to calculate the coefficients is by evaluation of:

$$c_{\beta\gamma}^{(\alpha)} = \frac{\beta!\,\gamma!}{(s-\beta)! + (s-\gamma)! + (s-\alpha)!} \tag{C.10}$$

where

$$s = \frac{\alpha+\beta+\gamma}{2} \tag{C.11}$$

in case $\alpha+\beta+\gamma$ is even and α, β and γ satisfy the triangle property (the sum of any two should not be greater than the third) [21]. If the condition is not met, the constant equals zero [21].

D Direct computation of the PCE coefficients of input parameters

The coefficients of a PC expansion of some input variable can be determined efficiently if the input variable and the uncertain variable are linearly related. Let x be the uncertain parameter with a log-normal distribution, directly related to the standard normal distributed variable ξ by:

$$x(\xi) = \exp(\mu + \sigma\xi) \tag{D.1}$$

where μ is the mean and σ is the standard deviation of the distribution of x. The polynomial chaos expansion of x is defined as:

$$x(\xi) \approx \sum_{\alpha} x^{(\alpha)} \Psi_\alpha(\xi) \tag{D.2}$$

where $x^{(\alpha)}$ are the coefficients of the expansion and $\Psi_\alpha(\xi)$ are the polynomials. An expression for the coefficients of the expansion can be obtained by projection the expansion onto $\Psi_\alpha(\xi)$:

$$\mathbb{E}\Big[x(\xi)\Psi_\alpha(\xi)\Big] = \mathbb{E}\Big[\sum_{\alpha} x^{(\alpha)} \Psi_\alpha(\xi)\Psi_\alpha(\xi)\Big] \tag{D.3}$$

which can be rewritten knowing the polynomials in the basis are orthonormal:

$$x^{(\alpha)} = \mathbb{E}\Big[x(\xi)\Psi_\alpha(\xi)\Big] = \int_{xi\in\Xi} x(\xi)\Psi_\alpha(\xi) f(\xi) d\xi \tag{D.4}$$

where Ξ is the span of the stochastic space and $f(\xi)$ is the probability distribution function of ξ [22]. The integral in (D.4) can be evaluated for different orders of α. For the first three orders the derivation is shown here. For the zeroth order holds:

$$x^{(0)} = \mathbb{E}\Big[e^{\mu+\sigma\xi}1\Big] = \int_{-\infty}^{\infty} e^{\mu+\sigma\xi}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\xi^2}d\xi = \frac{1}{\sqrt{2\pi}}e^{\mu+\frac{1}{2}\sigma^2}\int_{-\infty}^{\infty} e^{-\frac{1}{2}(\xi-\sigma)^2}d\xi = e^{\mu+\frac{1}{2}\sigma^2} \tag{D.5}$$

For the first order holds:

$$x^{(1)} = \mathbb{E}\Big[e^{\mu+\sigma\xi}\xi\Big] = \int_{-\infty}^{\infty} e^{\mu+\sigma\xi}\xi\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\xi^2}d\xi = \frac{1}{\sqrt{2\pi}}e^{\mu+\frac{1}{2}\sigma^2}\int_{-\infty}^{\infty} \xi e^{-\frac{1}{2}(\xi-\sigma)^2}d\xi = \sigma e^{\mu+\frac{1}{2}\sigma^2} \tag{D.6}$$

For the second order holds:

$$\begin{aligned} x^{(2)} &= \mathbb{E}\Big[e^{\mu+\sigma\xi}\Big(\frac{1}{2}\xi^2 - \frac{1}{2}\Big)\Big] = \int_{-\infty}^{\infty} e^{\mu+\sigma\xi}\Big(\frac{1}{2}\xi^2 - \frac{1}{2}\Big)\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\xi^2}d\xi \\ &= \frac{1}{\sqrt{2\pi}}e^{\mu+\frac{1}{2}\sigma^2}\int_{-\infty}^{\infty}\Big(\frac{1}{2}\xi^2 - \frac{1}{2}\Big)e^{-\frac{1}{2}(\xi-\sigma)^2}d\xi = \frac{1}{2}\sigma^2 e^{\mu+\frac{1}{2}\sigma^2} \end{aligned} \tag{D.7}$$

Similar derivations can be done to obtain the higher order coefficients. Because the distribution of the uncertain input variables is known, the relations (D.5), (D.6) and (D.7) can be evaluated to find the values of the coefficients. The zeroth order coefficient should approximate the mean of the distribution:

$$\mathbb{E}[x(\xi)] \approx x^{(0)} \tag{D.8}$$

and the higher order coefficients are related to the variance of the distribution by:

$$\mathrm{var}(x(\xi)) \approx \sum_{\alpha}(x^{(\alpha)})^2\alpha! \qquad \text{for } \alpha \neq 0 \tag{D.9}$$

[22].

If the uncertain parameter $x(\xi)$ does not directly appear as input parameter in the model, but is an input parameter for another variable in the model, the coefficients of the variable can be determined using the coefficients of $x(\xi)$. Let $A(x, y)$ be an variable which depends linearly on the uncertain parameter x and a certain parameter y. One can calculate the PCE coefficients of A according to:

$$A^{(0)} = A(x^{(0)}, y) \qquad A^{(1)} = A(x^{(1)}, y) \qquad \ldots \qquad A^{(n)} = A(x^{(n)}, y) \tag{D.10}$$

If y is also an uncertain variable $y(\xi)$ which is linearly related to A, the coefficients of both variables must be used to determine the coefficients of A:

$$A^{(0,0)} = A(x^{(0)}, y^{(0)}) \qquad A^{(1,0)} = A(x^{(1)}, 0) \qquad A^{(0,1)} = A(0, y^{(1)}) \tag{D.11}$$

$$A^{(2,0)} = A(x^{(2)}, 0) \qquad A^{(1,1)} = A(0, 0) \qquad A^{(0,2)} = A(0, y^{(2)}) \qquad \dots \tag{D.12}$$

E Matrices for the dynamic analysis of the slider crank

E.1 The FE mass and stiffness matrices

The FE mass matrix of one element in the bodies of the slider crank is defined as:

$$\boldsymbol{M}_{el} = \frac{\rho AL}{420}\begin{bmatrix} 140 & 0 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 0 & 0 \\ 0 & 156 & 0 & 0 & 0 & 22L & 0 & 54 & 0 & 0 & 0 & -13L \\ 0 & 0 & 156 & 0 & -22L & 0 & 0 & 0 & 54 & 0 & 13L & 0 \\ 0 & 0 & 0 & \frac{140I_x}{A} & 0 & 0 & 0 & 0 & 0 & -\frac{70I_x}{A} & 0 & 0 \\ 0 & 0 & -22L & 0 & 4L^2 & 0 & 0 & 0 & -13L & 0 & -\frac{3}{2}L^2 & 0 \\ 0 & 22L & 0 & 0 & 0 & 4L^2 & 0 & 13L & 0 & 0 & 0 & -\frac{3}{2}L^2 \\ 70 & 0 & 0 & 0 & 0 & 0 & 140 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 & 0 & 0 & 0 & 13L & 0 & 156 & 0 & 0 & 0 & -22L \\ 0 & 0 & 54 & 0 & -13L & 0 & 0 & 0 & 156 & 0 & 22L & 0 \\ 0 & 0 & 0 & -\frac{70I_x}{A} & 0 & 0 & 0 & 0 & 0 & \frac{140I_x}{A} & 0 & 0 \\ 0 & 0 & 13L & 0 & -\frac{3}{2}L^2 & 0 & 0 & 0 & 22L & 0 & 4L^2 & 0 \\ 0 & -13L & 0 & 0 & 0 & -\frac{3}{2}L^2 & 0 & -22L & 0 & 0 & 0 & 4L^2 \end{bmatrix} \tag{E.1}$$

where ρ is the density, A is the cross-sectional area of the element, L is the length of the element and I_x is the area moment of inertia about the local x-axis [23].

The FE stiffness matrix of one element in the bodies of the slider crank is defined as:

$$\boldsymbol{K}_{el} = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 & -a & 0 & 0 & 0 & 0 & 0 \\ 0 & 12b_z & 0 & 0 & 0 & 6Lb_z & 0 & -12b_z & 0 & 0 & 0 & 6Lb_z \\ 0 & 0 & 12b_y & 0 & -6Lb_y & 0 & 0 & 0 & -12b_y & 0 & -6Lb_y & 0 \\ 0 & 0 & 0 & c & 0 & 0 & 0 & 0 & 0 & -c & 0 & 0 \\ 0 & 0 & -6Lb_y & 0 & 4L^2b_y & 0 & 0 & 0 & 6Lb_y & 0 & 2L^2b_y & 0 \\ 0 & 6Lb_z & 0 & 0 & 0 & 4L^2b_z & 0 & -6Lb_z & 0 & 0 & 0 & 2L^2b_z \\ -a & 0 & 0 & 0 & 0 & 0 & a & 0 & 0 & 0 & 0 & 0 \\ 0 & -12b_z & 0 & 0 & 0 & -6Lb_z & 0 & 12b_z & 0 & 0 & 0 & -6Lb_z \\ 0 & 0 & -12b_y & 0 & 6Lb_y & 0 & 0 & 0 & 12b_y & 0 & 6Lb_y & 0 \\ 0 & 0 & 0 & -c & 0 & 0 & 0 & 0 & 0 & c & 0 & 0 \\ 0 & 0 & -6Lb_y & 0 & 2L^2b_y & 0 & 0 & 0 & 6Lb_y & 0 & 4L^2b_y & 0 \\ 0 & 6Lb_z & 0 & 0 & 0 & 2L^2b_z & 0 & -6Lb_z & 0 & 0 & 0 & 4L^2b_z \end{bmatrix} \tag{E.2}$$

where $a = \frac{EA}{L}$, $b_y = \frac{EI_y}{L^3}$, $b_y = \frac{EI_z}{L^3}$ and $c = \frac{GJ}{L}$, E is the Young's modulus, I_y is the area moment of inertia about the local y-axis, I_z is the area moment of inertia about the local z-axis, G is the shear modulus and J is the polar moment of inertia [23].

E.2 Derivatives of the constraints

The derivatives of the constraints of the slider crank with respect to the generalized coordinates are determined by hand to ensure the rotation matrices are not partitioned in the resulting matrices. The constraints are defined as:

$$\boldsymbol{\Phi}(\boldsymbol{q}, t) = \boldsymbol{0} \tag{E.3}$$

where $\boldsymbol{\Phi}$ is the vector of the constraints, $\boldsymbol{q}$ are the generalized coordinates and t is time. The first derivative of the constraints with respect to time is:

$$\dot{\boldsymbol{\Phi}}(\boldsymbol{q}, t) = \boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{\boldsymbol{t}} \tag{E.4}$$

where the subscript $\boldsymbol{q}$ denotes the derivative of the constraints to the generalized coordinates and the subscript t denotes the derivative directly to time. The matrix $\boldsymbol{\Phi}_{\boldsymbol{q}}$ is the Jacobian. The second derivative of the constraints with respect to time is:

$$\ddot{\boldsymbol{\Phi}}(\boldsymbol{q}, t) = [\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}]_{\boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{\boldsymbol{q}}\ddot{\boldsymbol{q}} + 2\boldsymbol{\Phi}_{\boldsymbol{q}_t}\dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{tt} \tag{E.5}$$

Note that for the slider crank time is not an explicit variable. Therefore, the derivative of the constraints with respect to time directly equals zero:

$$\boldsymbol{\Phi}_t = \mathbf{0}_{\{11\times 1\}} \tag{E.6}$$

as well as the second derivative with respect to time:

$$\boldsymbol{\Phi}_{tt} = \mathbf{0}_{\{11\times 1\}} \tag{E.7}$$

and the derivative of the Jacobian with respect to time:

$$\boldsymbol{\Phi}_{\boldsymbol{q}t} = \mathbf{0}_{\{11\times 12\}} \tag{E.8}$$

The Jacobian and the expression for $[\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}]_{\boldsymbol{q}}$ can be found by expanding (E.5) for each constraint. The derivation is similar for each translation and each rotation constraint and will therefore be shown only for the constraints of the slider crank at point A. To perform the derivation, one must know that the derivative of the rotation matrix to time is defined as:

$$\dot{\boldsymbol{R}} = \tilde{\boldsymbol{\omega}}\boldsymbol{R} \tag{E.9}$$

the product of a skew symmetric matrix $\tilde{\boldsymbol{a}}$ and a vector $\boldsymbol{b}$ can be written as:

$$\tilde{\boldsymbol{a}}\boldsymbol{b} = -\tilde{\boldsymbol{b}}\boldsymbol{a} \tag{E.10}$$

the transpose of a skew symmetric matrix can be written as:

$$\tilde{\boldsymbol{a}}^T = -\tilde{\boldsymbol{a}} \tag{E.11}$$

and a matrix expressed in frame O ($\boldsymbol{A}^O$) can be expressed in frame j ($\boldsymbol{A}^j$) using rotation matrices:

$$\boldsymbol{A}^j = \boldsymbol{R}_O^j \boldsymbol{A}^O \boldsymbol{R}_j^O \tag{E.12}$$

First, the translation constraint in point A is considered:

$$\boldsymbol{\Phi} = \boldsymbol{r}_1^{OO} + \boldsymbol{R}_1^O \boldsymbol{r}_A^{11} - \boldsymbol{r}_2^{OO} - \boldsymbol{R}_2^O \boldsymbol{r}_A^{22} \tag{E.13}$$

Using (E.9), the derivative with respect to time of the constraint can be written as:

$$\frac{\partial \boldsymbol{\Phi}}{\partial t} = \dot{\boldsymbol{r}}_1^{OO} + \tilde{\boldsymbol{\omega}}_1^{OO}\boldsymbol{R}_1^O\boldsymbol{r}_A^{11} + \boldsymbol{R}_1^O\dot{\boldsymbol{r}}_A^{11} - \dot{\boldsymbol{r}}_2^{OO} - \tilde{\boldsymbol{\omega}}_2^{OO}\boldsymbol{R}_2^O\boldsymbol{r}_A^{22} - \boldsymbol{R}_2^O\dot{\boldsymbol{r}}_A^{22} \tag{E.14}$$

and the second derivative as:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} =& \ddot{\boldsymbol{r}}_1^{OO} + \tilde{\boldsymbol{\alpha}}_1^{OO}\boldsymbol{R}_1^O\boldsymbol{r}_A^{11} + \tilde{\boldsymbol{\omega}}_1^{OO}\tilde{\boldsymbol{\omega}}_1^{OO}\boldsymbol{R}_1^O\boldsymbol{r}_A^{11} + 2\tilde{\boldsymbol{\omega}}_1^{OO}\boldsymbol{R}_1^O\dot{\boldsymbol{r}}_A^{11} + \boldsymbol{R}_1^O\ddot{\boldsymbol{r}}_A^{11} \\ & - \ddot{\boldsymbol{r}}_2^{OO} - \tilde{\boldsymbol{\alpha}}_2^{OO}\boldsymbol{R}_2^O\boldsymbol{r}_A^{22} - \tilde{\boldsymbol{\omega}}_2^{OO}\tilde{\boldsymbol{\omega}}_2^{OO}\boldsymbol{R}_2^O\boldsymbol{r}_A^{22} - 2\tilde{\boldsymbol{\omega}}_2^{OO}\boldsymbol{R}_2^O\dot{\boldsymbol{r}}_A^{22} - \boldsymbol{R}_2^O\ddot{\boldsymbol{r}}_A^{22}\end{aligned} \tag{E.15}$$

One can use (E.10), (E.11) and (E.12) to rewrite (E.15):

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} =& \ddot{\boldsymbol{r}}_1^{OO} - \boldsymbol{R}_1^O\tilde{\boldsymbol{r}}_A^{11}\boldsymbol{R}_O^1\boldsymbol{\alpha}_1^{OO} - \tilde{\boldsymbol{\omega}}_1^{OO}\boldsymbol{R}_1^O\tilde{\boldsymbol{r}}_A^{11}\boldsymbol{R}_O^1\boldsymbol{\omega}_1^{OO} + 2\tilde{\boldsymbol{\omega}}_1^{OO}\boldsymbol{R}_1^O\dot{\boldsymbol{r}}_A^{11} + \boldsymbol{R}_1^O\ddot{\boldsymbol{r}}_A^{11} \\ & - \ddot{\boldsymbol{r}}_2^{OO} + \boldsymbol{R}_2^O\tilde{\boldsymbol{r}}_A^{22}\boldsymbol{R}_O^2\boldsymbol{\alpha}_2^{OO} + \tilde{\boldsymbol{\omega}}_2^{OO}\boldsymbol{R}_2^O\tilde{\boldsymbol{r}}_A^{22}\boldsymbol{R}_O^2\boldsymbol{\omega}_2^{OO} - 2\tilde{\boldsymbol{\omega}}_2^{OO}\boldsymbol{R}_2^O\dot{\boldsymbol{r}}_A^{22} - \boldsymbol{R}_2^O\ddot{\boldsymbol{r}}_A^{22}\end{aligned} \tag{E.16}$$

The velocity and acceleration vectors of point A with respect to the floating frames and expressed in the floating frames can be expressed using flexible deformation shapes and flexible coordinates:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} =& \ddot{\boldsymbol{r}}_1^{OO} - \boldsymbol{R}_1^O\tilde{\boldsymbol{r}}_A^{11}\boldsymbol{R}_O^1\boldsymbol{\alpha}_1^{OO} - \tilde{\boldsymbol{\omega}}_1^{OO}\boldsymbol{R}_1^O\tilde{\boldsymbol{r}}_A^{11}\boldsymbol{R}_O^1\boldsymbol{\omega}_1^{OO} + 2\tilde{\boldsymbol{\omega}}_1^{OO}\boldsymbol{R}_1^O\boldsymbol{\Phi}_{1A}\dot{\boldsymbol{\eta}}_1 + \boldsymbol{R}_1^O\boldsymbol{\Phi}_{1A}\ddot{\boldsymbol{\eta}}_1 \\ & - \ddot{\boldsymbol{r}}_2^{OO} + \boldsymbol{R}_2^O\tilde{\boldsymbol{r}}_A^{22}\boldsymbol{R}_O^2\boldsymbol{\alpha}_2^{OO} + \tilde{\boldsymbol{\omega}}_2^{OO}\boldsymbol{R}_2^O\tilde{\boldsymbol{r}}_A^{22}\boldsymbol{R}_O^2\boldsymbol{\omega}_2^{OO} - 2\tilde{\boldsymbol{\omega}}_2^{OO}\boldsymbol{R}_2^O\boldsymbol{\Phi}_{2A}\dot{\boldsymbol{\eta}}_2 - \boldsymbol{R}_2^O\boldsymbol{\Phi}_{2A}\ddot{\boldsymbol{\eta}}_2\end{aligned} \tag{E.17}$$

which can be written in matrix vector form:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} = &\begin{bmatrix}\mathbf{1} & -\boldsymbol{R}_1^O \tilde{\boldsymbol{r}}_A^{11} \boldsymbol{R}_O^1 & \boldsymbol{R}_1^O \boldsymbol{\Phi}_{1A} & \mathbf{-1} & \boldsymbol{R}_2^O \tilde{\boldsymbol{r}}_A^{22} \boldsymbol{R}_O^2 & -\boldsymbol{R}_2^O \boldsymbol{\Phi}_{2A}\end{bmatrix} \begin{bmatrix}\ddot{\boldsymbol{r}}_1^{OO} \\ \boldsymbol{\alpha}_1^{OO} \\ \ddot{\boldsymbol{\eta}}_1 \\ \ddot{\boldsymbol{r}}_2^{OO} \\ \boldsymbol{\alpha}_2^{OO} \\ \ddot{\boldsymbol{\eta}}_2\end{bmatrix} \\ &+ \begin{bmatrix}\mathbf{0} & -\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \tilde{\boldsymbol{r}}_A^{11} \boldsymbol{R}_O^1 & 2\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{\Phi}_{1A} & \mathbf{0} & \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{r}}_A^{22} \boldsymbol{R}_O^2 & -2\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{\Phi}_{2A}\end{bmatrix} \begin{bmatrix}\dot{\boldsymbol{r}}_1^{OO} \\ \boldsymbol{\omega}_1^{OO} \\ \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{r}}_2^{OO} \\ \boldsymbol{\omega}_2^{OO} \\ \dot{\boldsymbol{\eta}}_2\end{bmatrix}\end{aligned} \tag{E.18}$$

where the first matrix is the Jacobian of the translation constraint in A and the second matrix is the $[\boldsymbol{\Phi}_{\boldsymbol{q}} \dot{\boldsymbol{q}}]_{\boldsymbol{q}}$ term of the translation constraint in A.

One of the rotation constraints in point A is expressed as:

$$\boldsymbol{\Phi} = (\boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\boldsymbol{R}_2^O \boldsymbol{n}_y) \tag{E.19}$$

of which the first derivative with respect to time can be determined using (E.9):

$$\frac{\partial \boldsymbol{\Phi}}{\partial t} = (\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\boldsymbol{R}_2^O \boldsymbol{n}_y) + (\boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y) \tag{E.20}$$

as well as the second derivative with respect to time:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} =&(\tilde{\boldsymbol{\alpha}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\boldsymbol{R}_2^O \boldsymbol{n}_y) + (\tilde{\boldsymbol{\omega}}_1^{OO} \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\boldsymbol{R}_2^O \boldsymbol{n}_y) + (\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y) \\ &+ (\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y) + (\boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\tilde{\boldsymbol{\alpha}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y) + (\boldsymbol{R}_1^O \boldsymbol{n}_z)^T (\tilde{\boldsymbol{\omega}}_2^{OO} \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y)\end{aligned} \tag{E.21}$$

By expanding the transposed terms and applying (E.11), one can write the second derivative of the rotation constraint as:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} =& - (\boldsymbol{n}_z)^T \boldsymbol{R}_0^1 \tilde{\boldsymbol{\alpha}}_1^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y + (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y \\ &- (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\alpha}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_2^{OO} \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{n}_y\end{aligned} \tag{E.22}$$

One can use (E.10), (E.11) and (E.12) to rewrite (E.22):

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} =&(\boldsymbol{n}_z)^T \boldsymbol{R}_0^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 \boldsymbol{\alpha}_1^{OO} - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 \boldsymbol{\omega}_1^{OO} + (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 \boldsymbol{\omega}_2^{OO} \\ &+ (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 \boldsymbol{\omega}_2^{OO} - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 \boldsymbol{\alpha}_2^{OO} - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 \boldsymbol{\omega}_2^{OO}\end{aligned} \tag{E.23}$$

which can again be expressed in matrix vector form:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Phi}}{\partial t^2} = &\begin{bmatrix}\mathbf{0} & (\boldsymbol{n}_z)^T \boldsymbol{R}_0^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 & \mathbf{0} & \mathbf{0} & -(\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 & \mathbf{0}\end{bmatrix} \begin{bmatrix}\ddot{\boldsymbol{r}}_1^{OO} \\ \boldsymbol{\alpha}_1^{OO} \\ \ddot{\boldsymbol{\eta}}_1 \\ \ddot{\boldsymbol{r}}_2^{OO} \\ \boldsymbol{\alpha}_2^{OO} \\ \ddot{\boldsymbol{\eta}}_2\end{bmatrix} \\ &+ \begin{bmatrix}\mathbf{0} & -(\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 & \mathbf{0} & \mathbf{0} & 2(\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 & \mathbf{0}\end{bmatrix} \begin{bmatrix}\dot{\boldsymbol{r}}_1^{OO} \\ \boldsymbol{\omega}_1^{OO} \\ \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{r}}_2^{OO} \\ \boldsymbol{\omega}_2^{OO} \\ \dot{\boldsymbol{\eta}}_2\end{bmatrix}\end{aligned} \tag{E.24}$$

where the first matrix is the Jacobian of the rotation constraint in A and the second matrix is the $[\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}]_{\boldsymbol{q}}$ term of the rotation constraint in A.

By performing the derivation of the Jacobian and $[\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}]_{\boldsymbol{q}}$ for every constraint, the complete Jacobian and $[\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}]_{\boldsymbol{q}}$ matrices can be constructed. The Jacobian of the slider cranks is:

$$\boldsymbol{\Phi}_{\boldsymbol{q}} = \begin{bmatrix}
1 & 0 & 0 & \left[-\boldsymbol{R}_O^1 \tilde{\boldsymbol{r}}_O^{11} \boldsymbol{R}_O^1 \right] & \left[\boldsymbol{R}_1^O \boldsymbol{\Phi}_{1O}\right] & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\
0 & 0 & 1 & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\
0 & 0 & 0 & [(\boldsymbol{n}_y)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{n}}_z] & 0 \ \ldots \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\
1 & 0 & 0 & \left[-\boldsymbol{R}_O^1 \tilde{\boldsymbol{r}}_O^{11} \boldsymbol{R}_O^1 \right] & \left[\boldsymbol{R}_1^O \boldsymbol{\Phi}_{1A}\right] & -1 & 0 & 0 & \left[\boldsymbol{R}_O^2 \tilde{\boldsymbol{r}}_O^{22} \boldsymbol{R}_O^2 \right] & & \left[-\boldsymbol{R}_2^O \boldsymbol{\Phi}_{2A}\right] & & \\
0 & 1 & 0 & & & 0 & -1 & 0 & & & & & & \\
0 & 0 & 1 & & & 0 & 0 & -1 & & & & & & \\
0 & 0 & 0 & [(\boldsymbol{n}_z)^T \boldsymbol{R}_0^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & 0 \ \ldots \ 0 & 0 & 0 & 0 & [-(\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & & 0 & \ldots & 0 \\
0 & 0 & 0 & [(\boldsymbol{n}_x)^T \boldsymbol{R}_0^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & 0 \ \ldots \ 0 & 0 & 0 & 0 & [-(\boldsymbol{n}_x)^T \boldsymbol{R}_O^1 \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & & 0 & \ldots & 0 \\
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0 & 0 & 1 & 0 & \left[-\boldsymbol{R}_O^2 \tilde{\boldsymbol{r}}_O^{22} \boldsymbol{R}_O^2 \right]_{\text{row2\&3}} & & \left[\boldsymbol{R}_2^O \boldsymbol{\Phi}_{2A}\right]_{\text{row2\&3}} & & \\
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0 & 0 & 0 & 1 & & & & &
\end{bmatrix} \tag{E.25}$$

The $[\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}]_{\boldsymbol{q}}$ matrix of the slider crank is:

$$[\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}]_{\boldsymbol{q}} = \begin{bmatrix}
0 & 0 & 0 & \left[-\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_O^1 \tilde{\boldsymbol{r}}_O^{11} \boldsymbol{R}_O^1 \right] & \left[2\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{\Phi}_{1O}\right] \\
0 & 0 & 0 & & \\
0 & 0 & 0 & & \\
0 & 0 & 0 & [-(\boldsymbol{n}_y)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \tilde{\boldsymbol{n}}_z] & 0 \ \ldots \ 0 \\
0 & 0 & 0 & \left[-\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_O^1 \tilde{\boldsymbol{r}}_O^{11} \boldsymbol{R}_O^1 \right] & \left[2\tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_1^O \boldsymbol{\Phi}_{1A}\right] \\
0 & 0 & 0 & & \\
0 & 0 & 0 & & \\
0 & 0 & 0 & [-(\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & 0 \ \ldots \ 0 \\
0 & 0 & 0 & [-(\boldsymbol{n}_x)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & 0 \ \ldots \ 0 \\
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0 \\
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0
\end{bmatrix}$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0 \\
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0 \\
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0 \\
0 & 0 & 0 & 0 \ 0 \ 0 & 0 \ \ldots \ 0 \\
0 & 0 & 0 & \left[\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_O^2 \tilde{\boldsymbol{r}}_O^{22} \boldsymbol{R}_O^2 \right] & \left[-2\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{\Phi}_{2A} \right] \\
0 & 0 & 0 & & \\
0 & 0 & 0 & & \\
0 & 0 & 0 & [2(\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 - (\boldsymbol{n}_z)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & 0 \ \ldots \ 0 \\
0 & 0 & 0 & [2(\boldsymbol{n}_x)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_1^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2 - (\boldsymbol{n}_x)^T \boldsymbol{R}_O^1 \tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \tilde{\boldsymbol{n}}_y \boldsymbol{R}_O^2] & 0 \ \ldots \ 0 \\
0 & 0 & 0 & \left[-\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_O^2 \tilde{\boldsymbol{r}}_O^{22} \boldsymbol{R}_O^2 \right]_{\text{row2\&3}} & \left[2\tilde{\boldsymbol{\omega}}_2^{OO} \boldsymbol{R}_2^O \boldsymbol{\Phi}_{2B}\right]_{\text{row2\&3}} \\
0 & 0 & 0 & & \\
\end{bmatrix} \tag{E.26}$$